On Feedback for Network Coding

Christina Fragouli, Desmond Lun, Muriel Médard, Payam Pakzad

christina.fragouli@epfl.ch, dslun@mit.edu, medard@mit.edu, payam@digitalfountain.com

***Abstract*— In this paper we examine possible ways that feedback can be used, in the context of systems with network coding capabilities. We illustrate, through a number of simple examples, that use of feedback can be employed for parameter adaptation to satisfy QoS requirements as well as for reliability purposes. We also argue that there are benefits in applying network coding to the feedback packets themselves, and finally, we examine the design of acknowledgment packets.**

I. Introduction

In this paper we present a first attempt to study feedback in the context of network coding. The questions we are interested in are 1) how can feedback help, and 2) how should feedback be designed. More generally, we look at the problem of designing and using control information in a system that employs network coding. In this context, we propose a number of ideas, that we think mature, are promising directions for further research.

We consider networks that can be represented as directed graphs, with each edge corresponding to a packet erasure channel. A source located at a vertex of the graph multicasts information to N receivers $R_1, R_2, \ldots, R_N$. For $N = 1$ we get a unicast connection as a special case.

Let m_i denote the information-theoretic min-cut capacity between the source and receiver R_i, and let $m := \min_i m_i$. From information-theoretic arguments we know that the source can multicast information to each receiver at a rate arbitrarily close to m, by allowing intermediate nodes in the network to randomly combine their incoming information streams [1]. This result assumes that the source knows the min-cut value and encodes information at the corresponding information rate. Moreover, it assumes no complexity constraints: each intermediate node in the network encodes its incoming information streams using an arbitrarily complex forward error correction (FEC) scheme.

FEC schemes that attempt to approach this performance under more practical constraints were previously investigated for example in [2], [3], [4], [7]. These schemes operate at a network or application layer, and employ low complexity processing at intermediate nodes.

Feedback (e.g., ARQ) offers an alternative to FEC. Idealized feedback assumes that a transmitter instantaneously knows whether a transmitted packet has been successfully received or not. In the case of a unicast connection over a network of erasure channels, idealized feedback allows to achieve the min-cut capacity with no coding at the transmitter. Feedback is also used for congestion control and rate adaptation. In the context of network coding such mechanisms are explored in [13], [14], [15].

In practice, feedback comes at a cost. Acknowledgment packets form a new source of traffic in the network that needs to be reliably transmitted. Moreover, waiting for acknowledgment packets to arrive may increase delay. Clearly, there are interesting trade-offs between the use of feedback and forward error correction for network coded schemes to be explored.

In this paper we outline a number of examples, illustrating the benefits that the use of feedback can offer for network coded traffic (Section II), the benefits of applying network coding to the feedback packets themselves (Section III), and the benefits of designing acknowledgment packets specifically for network coded traffic (Section IV). detail.

II. How Can Feedback Help

Current networking protocols employ feedback towards two goals: reliability against packet losses, and rate adaptation. We argue in this section, and illustrate through simple examples, that using appropriately designed feedback one can achieve similar goals for systems that employ network coding.

A. Parameter Adaptation

In a network where the available link capacities dynamically change – for example due to varying traffic patterns – feedback can be used to

adapt network coding parameters to satisfy QoS requirements. As a specific instance, consider an application multicasting packets to a set of receivers, over a network subject to packet erasures. Assume an asynchronous network operation, where intermediate nodes uniformly at random combine packets. In practical implementations of network coding, the information packets are divided into sets called generations, and intermediate nodes on the network linearly combine only packets belonging in the same generation [5].

As is observed from experimental studies in the literature [6], the size of the generation affects the achievable rate the receivers observe. Indeed, it affects the size of the erasure correcting codes employed and thus the erasure correcting capabilities. Moreover, it also affects the multicasting capacity, i.e., whether the average min-cut rate towards all receivers in the multicast group is achieved.

Assume now that we are interested in a delay-sensitive application, where packets decoded after a "deadline" are useless and are simply discarded. Use of a large generation size may allow to achieve higher information rates, thus reducing the average packet decoding time. On the other hand, use of a large generation size may increase the decoding delay, because receivers need to wait for the reception of more coded packets before decoding. Indeed, the generation size determines the size of decoding matrices the receivers use to decode the data, and thus, the block decoding delay is directly proportional to the size of the generation. We can summarize this trade-off as follows: using small generation sizes may reduce the throughput and erasure-correcting benefits of mixing information packets, while large generation sizes may incur unacceptable decoding delay at the receivers.

Feedback from the receivers to the source can be used to adaptively adjust the generation size, and thus maximize the number of packets successfully decoded within the delay specifications. The source response to this type of feedback can be thought of as being similar in nature to the TCP window, used by TCP for purposes of congestion control: closing the transmission window would correspond to reducing the generation size, while opening the transmission window would correspond to increasing the generation size. We illustrate this approach for the following simple example.

Example 1: Consider a combination network with h sources, N receivers, and k coding edges $\{A_iB_i\}$ as depicted in Fig. 1. Assume that time is slotted, and that at every time slot we can send one symbol through every edge (unit capacity edges). Moreover, assume that the edges between nodes A and B have an associated delay. This delay is a random variable distributed according to a given distribution $\mathcal{P}$, and operating using a "slower" clock. For example, assume that the delays associated with the links change in an i.i.d. fashion every M time-slots. Finally, due to QoS requirements, the receivers need to decode the data within a delay of $\mathcal{T}^{thres}$.

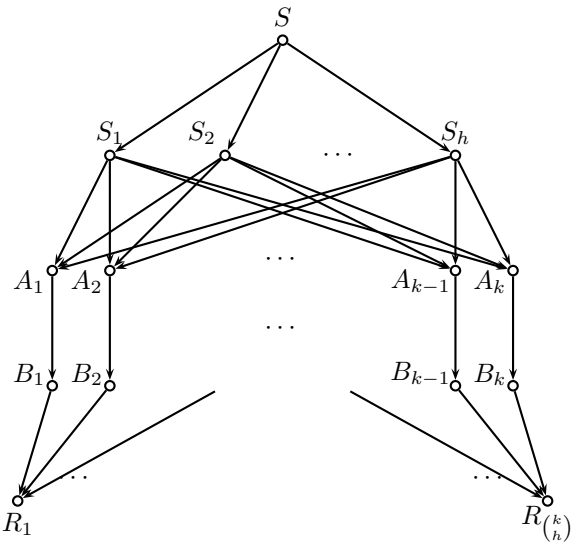

Fig. 1. The combination $B(h,k)$ network has h sources and $N = \binom{k}{h}$ receivers. The k nodes $\{A_i\}$ have mincut h from the source. Each receiver observes a distinct subset of h B-nodes, and thus has also mincut h. Edges have unit capacity and sources have unit rate.

To transmit at rate h to all receivers, the source can send at most h uncoded data packets through h edges AB, and additionally, $k-h$ linear combinations that depend on all source symbols through the remaining edges. Consider a receiver that observes h of these $k-h$ edges. Ignoring the processing delay, for the receiver to decode the data within a delay of $\mathcal{T}^{thres}$, *all* edges AB it observes need to incur delay less than $\mathcal{T}^{thres}$, which occurs with probability $\mathcal{P}(d < \mathcal{T}^{thres})^h$. Thus, if even one of the h edges incurs non-acceptable delay, all the h received symbols become useless to the receiver.

On the other extreme, if the source sends uncoded transmission through every edge, then a receiver does not need to perform decoding: an edge with delay greater than $\mathcal{T}^{thres}$ will not render any other received symbol useless. However, with uncoded transmissions there will exist receivers that observe

rate one (instead of h).

Without feedback, the source will in general choose the strategy that will with high probability maximize the achievable throughput within the delay constraints.

Assume now that the receivers send feedback to the source. The source can then simply not use the edges AB that incur unacceptable delay, and employ a coding scheme that allows to maximize the achievable rate using the remaining edges. To conclude, use of feedback may allow to optimize the achievable rate subject to rigid delay requirements. ♦

B. Reliability

We here argue that use of feedback can help to save resources, such as memory, in systems that provide reliable transmission over erasure networks. We also propose the use of schemes that combine ARQ and FEC at a given designed rate as described in the following.

Consider the simple example of a source A transmitting information to a destination C over a path with intermediate node B. Packets are dropped on paths AB and BC with probability ϵ_{AB} and ϵ_{BC} respectively. We assume that $\epsilon_{AB}\epsilon_{BC}$, and thus the min-cut capacity is $(1-\epsilon_{AB})$.

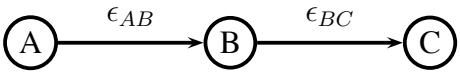

Fig. 2. A path from source A to receiver C through the intermediate node B.

Table I gives a small summary of schemes which we discuss in terms of delay, achievable rate and memory requirements, defined in the following.

The first scheme assumes that all nodes A, B and C employ a capacity-achieving FEC scheme, as described in [2], [3]. The source node A encodes k symbols to create n_1 coded outputs using a code C_1 and sends them over the channel AB. Node B will receive on average $n_1(1-\epsilon_{AB})$ coded symbols over n_1 time slots. Node B will send n_2 packets, using a code (more generally, processing) 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 $n_2(1-\epsilon_{BC})$ packets after d time slots. We use, similar to [4], the following metrics:
1) *Delay* incurred at the intermediate node B: this is the time $(d - k/C_{\mathrm{mc}})$, where C_{mc} is the min-cut capacity. It represents the excess time, relative to the theoretical minimum, that it takes for k packets to be communicated, disregarding any delay due to the use of the feedback channel. This quantity is similar to the overhead in the literature on rateless erasure codes (see, e.g., [8], [9], [10]).
2) *Blocksize*: the size of an individual coding block. This quantity is a reflection of delay as it is more conventionally referred to in networking literature, i.e., the time between when a single, particular packet is required at node A and when it is decoded, and available for use, at node C. Larger coding blocks correspond directly to larger values of this delay. As an example, for streaming applications where the data packets need to be consumed in the correct order and at the code rate, the block size immediately translates to the start-up delay before the information can be used without further interruption.
3) *Feedback*: the number of feedback packets used. This value proportionally contributes to the overall decoding delay.
4) *Memory requirement*: the number of memory elements needed at node B.
5) *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 component code achieves capacity over its corresponding channel.
Scheme II (end-to-end FEC) assumes that the intermediate node B simply forwards its incoming packets, and does not have the capability of more sophisticated storage and processing. Scheme III, that was investigated in [7], assumes that the coding node B has a fixed, finite memory of length m in which it stores packets formed from an incoming packet stream, and it sends packets formed from random linear combinations of its memory contents. This scheme achieves a factor $(1-\pi_m)$ of the min-cut capacity, where $\pi_m < \epsilon_{BC}$ is the steady state probability of an appropriately defined Markov chain as described in [7].

Alternatively to FEC, ARQ also allows to achieve the min-cut capacity. Current ARQ schemes operate on the assumption that we want to successfully receive all transmitted packets, and ensures that by using one feedback packet per message packet. We describe this as having feedback of rate $R_{\mathrm{f}} = 1$. On the other hand, in the case where we implement FEC using rateless codes [8], [9], we have one feedback

Schemes		Rate	Delay	Feedback	Memory	Blocksize
I	FEC	min-cut: $(1-\epsilon_{AB})$	$O(\sqrt{k\log k})$	0	$O(k)$	k
II	end-to-end FEC	$(1-\epsilon_{AB})(1-\epsilon_{BC})$	$O(k)$	0	0	k
III	FEC-m	$(1-\epsilon_{AB})(1-\pi_m)$	$O(k)$	0	m	k
IV	ARQ ($R_{\rm f}=1$)	min-cut: $(1-\epsilon_{AB})$	$O(\sqrt{k})$	k	$O(\sqrt{k})$	1
V	FEC+ARQ ($R_{\rm f}$)	min-cut: $(1-\epsilon_{AB})$	$O(\sqrt{k})$	$kR_{\rm f}$	$O(\sqrt{k})$	k

TABLE I

SCHEMES FOR PACKET ERASURE NETWORKS. FOR SCHEMES REQUIRING FEEDBACK, WE ASSUME THAT FEEDBACK IS LOSSLESS AND INSTANTANEOUS. DETAILED CALCULATIONS APPEAR IN [4].

message at the end of transmission (signifying "stop transmission"). Assume that we send k information packets. In this case, we say that we have feedback at an average rate of $1/(k(1+\epsilon))$. That is, the destination sends one feedback message as soon as it receives $k(1+\epsilon)$ packets and is thus able to decode the k information packets.

More generally, we define the feedback rate of a scheme as

$$R_{\rm f} = \frac{\text{number of feedback messages}}{\text{number of received packets}}.$$

Scheme V captures this general case. The scheme uses ARQ with rate $R_{\rm f} = 1/f$, i.e., every f time slots node C sends a feedback to node B indicating how many packets it has received. Node B can then release that many elements from its buffer. relaxed, for example, node C sends a feedback once it receives f packets, etc).

Note that the feedback delay for this scheme is smaller than the usual ARQ (with $R_{\rm f}=1$) by a factor of $R_{\rm f}$. At the same time, the average memory used at node B is only a constant $1/R_f(1-\epsilon_{BC})$ elements larger than that of ARQ. The main benefit of this scheme is that it allows us to achieve the min-cut rate, while keeping the average memory requirements at node B finite. Moreover, feedback is required only on link BC, as opposed to both links AB and BC. Memory requirements may become an important consideration in practical systems. For example, router B may have a common pool of memory, to be shared among several connections. Scheme V would allow more connections to benefit from processing at the router B.

III. APPLYING NETWORK CODING TO ACKNOWLEDGMENT PACKETS

We now argue there might exist strong incentives for applying network coding to the feedback packets themselves. In particular, it may lead to a significant reduction in terms of bandwidth, as the following example illustrates.

Consider multicasting along a tree, where N receivers, at the leaves of the tree, send acknowledgment packets back to the source at the root of the tree. These N ack packets attempting to reach the same destination may lead to congestion at the higher levels of the tree.

packets, The main overhead of the ack packets comes from the fact that they do not only contain the payload, i.e., the useful information, but also the overhead of the packet headers. Our observation is that, by slightly increasing the payload of the ack packets, and using network coding techniques, we can ensure that every edge in the tree is traversed by at most one ack packet.

Assume that the payload of the ack vector transmitted by receiver i is a length N vector, having one at position i and zero everywhere else. Each intermediate node, upon reception of multiple ack packets, XORs these vectors, and sends the resulting vector to its next ancestor. The procedure is repeated, until the root receives a single vector, with one at the position of receivers having successfully received the information. This procedure is illustrated in Fig. 3 for a particular configuration. Note that, intermediate nodes do not need to keep state information in order to implement this scheme. As soon as a single acknowledgment packet arrives at an intermediate node, it can trigger a time window associated with a particular multicasted information packet. The node then accumulates ack packets for the specific information packet during this window, and sends their XOR to its ancestors. This approach to concatenating acknowledgment packets is inspired by the inverse multicast tree monitoring approach in [11].

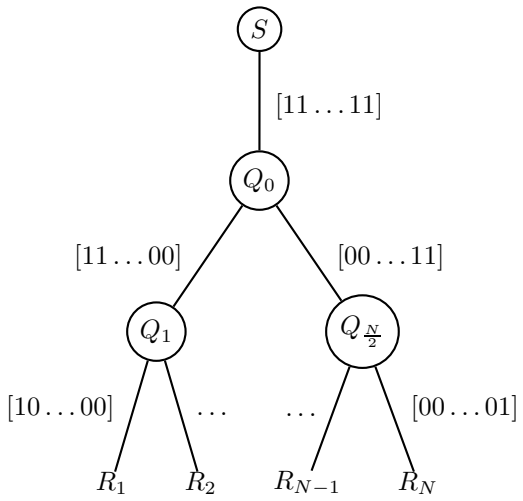

Fig. 3. A tree of depth two spanning N receivers and the associated ack packets.

IV. Acknowledgment Packet Design

In this section we look at the contents of ARQ packets that support network coded connections.

Consider again the unicast connection depicted in Fig. 2 where node A transmits h information packets to node C. The receiver C needs to receive h linear independent combinations of the source data. As is well known, *any* set of h linearly independent combinations, i.e., any basis of the h-dimensional space, would allow the receiver to decode. Thus, the receiver simply needs to acknowledge to the node B how many linear independent combinations (degrees of freedom) it has received, as opposed to the exact packets.

In particular, TCP assigns sequence numbers to bytes transmitted, and depending whether it receives an acknowledgment packet from the receiving host within a timeout interval, it either discards a packet or retransmits it. A receiver acknowledges the last received byte, using a sequence number of length 32 bits.

Acknowledging degrees of freedom allows to dispense of the sequence numbering the receiving host simply needs to acknowledge the number of received packets. Note that with network coding, assuming the information packets are generated over a large field, the probability that any received packet is a linear combination of the previously received packets is negligible (see [12]). Thus, the number of received packets will accurately reflect the number of degrees of freedom. It is clear that we can convey the number of received packets within the TCP prescribed timeframes with a much smaller number than 32 bits, for example, using 8 bits. Moreover, acknowledging degrees of freedom allows to reduces the overhead of keeping track of the sequences of packets received, which reduces both the computational and memory requirements.

V. Conclusions

In this paper we presented a number of simple observations that indicate the possible benefits use of feedback can offer in the context of network coding. Such benefits include, adaptive parameter optimization to satisfy QoS requirements, and reduction of resources such as memory elements and bandwidth consumption.

References

[1] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network information flow," *IEEE Trans. Inform. Theory*, vol. 46, no. 4, pp. 1204–1216, July 2000.

[2] D. Lun, M. Médard,M. Effros, "On coding for reliable communication over packet networks," in *Proc. Allerton Conference on Communication, Control, and Computing*, 2004.

[3] D. S. Lun, M. Médard, R. Koetter, and M. Effros. "Further results on coding for reliable communication over packet networks," in *Proc. ISIT 2005*

[4] P. Pakzad, C. Fragouli and A. Shokrollahi, "Coding schemes for line networks," in *Proc. ISIT 2005*.

[5] P. A. Chou, Y. Wu, and K. Jain, "Practical network coding", in *Proc. 41st Allerton Conf. Communication, Control and Computing*, Monticello, IL, Oct. 2003.

[6] C. Fragouli, J. Widmer and J.Y. LeBoudec, "Efficient broadcasting using network coding", *to appear in ACM Trans. on Networking*, 2007.

[7] D. S. Lun, P. Pakzad, C. Fragouli, M. Médard, and R. Koetter, "An analysis of finite-memory random linear coding on packet streams," in *Proc. WiOpt '06*.

[8] M. Luby, "LT codes," in Proc. *IEEE Symposium on the Foundations of Computer Science (STOC)*, 2002, pp. 271-280.

[9] A. Shokrollahi "Raptor codes", *IEEE Transactions on Information Theory*, vol. 52, iss. 6, pp. 2551-2567, 2006.

[10] P. Maymounkov, N. J. A. Harvey, and D. S. Lun. "Methods for efficient network coding," in *Proc. Allerton 2006*.

[11] C. Fragouli and A. Markopoulou, "A network coding approach to network monitoring," in *Proc. Allerton 2005*.

[12] T. Ho, R. Koetter, M. Médard, M. Effros, J. Shi, and D. Karger, "A random linear network coding approach to multicast," *IEEE Transactions on Information Theory*, vol. 52, iss. 10, pp. 4413-4430, October 2006.

[13] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," *in Proc. 43th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2005.

[14] T. Ho, Y.-H. Chang, and K. Han, "On constructive network coding for multiple unicasts", *in Proc. 44th Annual Allerton Conference on Communication, Control, and Computing*, Sept. 2006.

[15] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," *In Proc. 2007 Information Theory and Applications Workshop (ITA 2007)*, January-February 2007.