International Symposium on Information Theory and its Applications, ISITA2008
Auckland, New Zealand, 7-10, December, 2008

Online network coding for optimal throughput and delay – the three-receiver case

Jay Kumar Sundararajan, Devavrat Shah, Muriel Médard

Abstract

For a packet erasure broadcast channel with three receivers, we propose a new coding algorithm that makes use of feedback to dynamically adapt the code. Our algorithm is throughput optimal, and we conjecture that it also achieves an asymptotically optimal average decoding delay at the receivers. We consider heavy traffic asymptotics, where the load factor ρ approaches 1 from below with either the arrival rate (λ) or the channel parameter (μ) being fixed at a number less than 1. We verify through simulations that our algorithm achieves an asymptotically optimal decoding delay of $O\left(\frac{1}{1-\rho}\right)$.

1. Introduction

Reliable communication over packet erasure channels is a well studied problem. Several solutions have been proposed, each with its own requirements, merits and issues. In this work, we consider a three-receiver packet erasure broadcast channel with feedback and address questions of throughput and decoding delay at the receivers.

To communicate over a packet erasure broadcast channel, one can use the random linear network coding solution of [1], where the sender transmits a random linear combination of all packets that have arrived so far. Digital fountain codes ([2,3]) form another approach to this problem. These solutions use coding to ensure that with high probability, the transmitted packet will have what we call the **innovation guarantee property**, *i.e.*, it will be *innovative*[1] to every receiver that receives it, except if the receiver already knows as much as the sender. Thus, every successful reception brings a unit of new information. Such schemes achieve 100% throughput.

However, both fountain codes and random linear network coding perform block-based encoding. In general, the receiver may not be able to extract the original packets from the coded packets till the entire block has been received. This leads to a decoding delay, which is a problem for real-time packet streaming applications such as video. Ideally we want a code that would allow playback even before the full block is received. In other words, we are interested in minimizing the average per-packet delay. Related questions have been studied by [4], [5], [6] and [7].

With full feedback, the optimal scheme over a point-to-point packet erasure channel is Automatic Repeat reQuest (ARQ) – the sender simply retransmits a packet upon erasure. This scheme also has the advantage of being a sliding window approach as opposed to a block-based approach. Although it achieves 100% throughput and optimal packet delay, ARQ does not extend to broadcast-mode links. On the other hand, network coding readily extends to broadcast-mode links.

Reference [8] proposes a scheme that uses feedback to acknowledge degrees of freedom instead of original packets, thus combining the benefits of network coding and ARQ. This new framework allows the sender to dynamically adapt its code to incorporate receivers' states of knowledge. This fact was used to design a queue management algorithm called the drop-when-seen algorithm that minimizes the sender's queue size, along with a coding module that provides 100% throughput. Another related reference is [9], where the authors combine an acknowledgment scheme with network coding. Here, the main focus is to maximize the throughput. In contrast to these works, our current work focuses on achieiving the best possible decoding delay for all receivers, while maintaining optimal throughput.

By decoding delay of a receiver, we mean the time elapsed between the arrival of a packet into the sender's queue and its getting decoded by the receiver under consideration, averaged over the packets in the long run in a packet streaming scenario. This is different from but related to the notion of delay used in [7].

For the special case of a packet erasure broadcast channel with only two receivers, reference [10] proposes a feedback-based coding algorithm that not only achieves 100% throughput, but also guarantees that every successful innovative reception will cause the receiver to decode a new packet. We call this property *instantaneous decodability.* Instantaneous decodability and 100% throughput are both desirable goals. However, this approach does not extend to the case of more than two receivers. With prior knowledge of the erasure pattern, [7] gives an offline algorithm that achieves optimal delay and throughput for the case of three receivers.

The authors are in the Department of Electrical Engineering and Computer Science, at the Massachusetts Institute of Technology, Cambridge, MA 02139. Email: {jaykumar, devavrat, medard}@mit.edu. This material is based upon work under subcontract # 18870740-37362-C issued by Stanford University and supported by DARPA, and upon work under a subcontract # 060786 issued by BAE Systems National Security Solutions, Inc. and supported by DARPA and the Space and Naval Warfare System Center (SPAWARSYSCEN), San Diego under Contract No. N66001-06-C-2020.

[1]An innovative packet is a coded packet whose coefficient vector is outside the span of previously received packets' coefficient vectors.

However, in the online case, even with only three receivers, [10] shows through an example that it is not possible to simultaneously guarantee instantaneous decodability as well as throughput optimality.

In the light of this example, our current work aims for a relaxed version of instantaneous decodability while still retaining the requirement of optimal throughput. Our relaxation of the condition is as follows. Let λ and μ be the arrival rate and the channel quality parameter respectively. Let $\rho \triangleq \lambda/\mu$ be the load factor. We consider asymptotics when the load factor on the system tends to 1 (*i.e.*, 100%), while keeping either λ or μ fixed at a number less than 1. The optimal throughput requirement means that the queue of undelivered packets is stable for all values of ρ less than 1. Our new requirement on decoding delay is that the growth of the average decoding delay as $\rho \to 1$ should be the same as for the single receiver case. The expected per-packet delay of a receiver in a system with more than one receiver is clearly lower bounded by the corresponding quantity for a single-receiver system. Thus, instead of optimal decoding delay, we aim to guarantee asymptotically optimal decoding delay. The motivation is that in most practical systems, delay becomes a critical issue only when the system starts approaching its full capacity. When the load on the system is well within its capacity, the delay is usually small and hence not an issue. For the case of two receivers, it can be shown that this relaxed requirement is satisfied by the scheme in [10] due to the instantaneous decodability property, *i.e.*, the scheme achieves the asymptotically optimal average decoding delay per packet for the two-receiver case.

In our current work, we provide a new coding module for the case of three receivers that achieves optimal throughput. We conjecture that at the same time it also achieves an asymptotically optimal decoding delay in the following sense. With a single receiver, the optimal scheme is ARQ with no coding and we show that this achieves an expected per-packet delay at the sender of $\Theta\left(\frac{1}{1-\rho}\right)$. For the three-receiver system, we conjecture that our scheme also achieves a delay of $O\left(\frac{1}{1-\rho}\right)$, and thus meets the lower bound in an asymptotic sense. We have verified this behavior through simulations for values of ρ that are very close to 1. Our scheme thus achieves feedback-based control of the decoding delay, along the lines suggested in [11]. We believe that our approach can be extended to an arbitrary number of receivers as well.

2. Preliminaries

2.1. The setup

The setup is the same as in [8]. Time is slotted. Packets arrive into the sender's queue according to a Bernoulli process of rate λ. The sender wants to broadcast this stream to three receivers over a packet erasure broadcast channel.

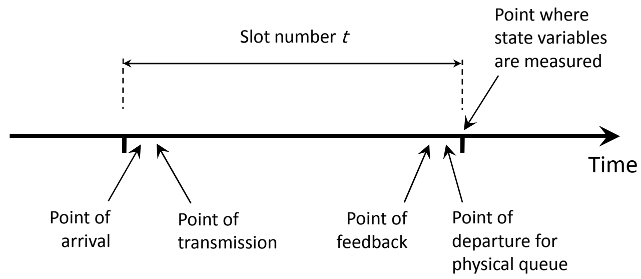
Slot number t
Point where state variables are measured
Time
Point of arrival
Point of transmission
Point of feedback
Point of departure for physical queue

Figure 1: Relative timing of arrival, service and departure points within a slot

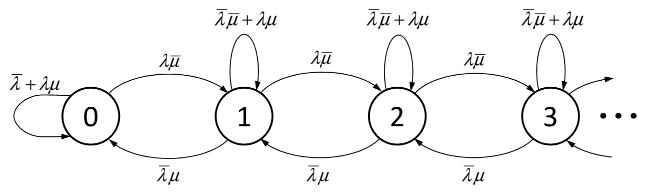
$\bar{\lambda}+\lambda\mu$
$\lambda\bar{\mu}$
$\bar{\lambda}\bar{\mu}+\lambda\mu$
$\bar{\lambda}\mu$
0
1
2
3
...

Figure 2: Markov chain for the sender's queue size – single receiver case. Here $\bar{\lambda} := (1-\lambda)$ and $\bar{\mu} := (1-\mu)$.

The sender has one queue with no preset size constraints. The channel accepts one packet per slot. Each receiver either receives this packet with no errors (with probability μ) or an erasure occurs (with probability $(1-\mu)$). Erasures occur independently across receivers and across slots and can be detected by receivers. There is perfect feedback in each slot. Figure 1 shows the timing of events in a slot. In particular, we assume that the sender finds out whether the receivers received the previous slot's transmission before selecting the current slot's transmission.

2.2. The lower bound

The expected per-packet delay for the single receiver case is clearly a lower bound for the corresponding quantity at one of the receivers in a three-receiver system. We will compute this lower bound in this section. Figure 2 shows the Markov chain for the queue size. If $\rho := \frac{\lambda}{\mu} < 1$, then the chain is positive recurrent and its steady state distribution can be found ([12]). Based on this, the steady state expected queue size can be computed to be $\frac{\rho(1-\mu)}{(1-\rho)} = \Theta\left(\frac{1}{1-\rho}\right)$. Now, if $\rho < 1$, then the system is stable and Little's law can be applied to show that the expected per-packet delay in the single receiver system is also $\Theta\left(\frac{1}{1-\rho}\right)$.

2.3. Representing knowledge

We treat packets as vectors over some finite field. Throughout this paper, we consider a single source that generates a stream of packets. The k^{th} packet that the source generates is said to have an *index* k and is denoted as $\mathbf{p_k}$. We assume that the sender uses only linear codes, *i.e.*, the transmission is some linear combination of packets. The linear combination is uniquely specified using the vector of coefficients used to form it. With linear codes, the state of knowledge of a node after receiving some set of linear combinations has a vector space structure. This is because the node can compute any linear combination whose coefficient vector is within the linear span of the coefficient vectors of previously received linear combinations. This leads to the following definition of *knowledge space* which we restate from [8].

Definition 1 (Knowledge of a node) *The* knowledge of a node *is the set of all linear combinations of original packets that it can compute, based on the information it has received so far. The coefficient vectors of these linear combinations form a vector space called the* knowledge space *of the node. The dimension of this vector space is called the* rank *of the node.*

We next restate the definition of a node "seeing" a message packet from [8]. A node is said to have *seen* a message packet $\mathbf{p}$ if it has received enough information to compute a linear combination of the form $(\mathbf{p} + \mathbf{q})$, where $\mathbf{q}$ is itself a linear combination involving only packets with an index greater than that of $\mathbf{p}$. (Decoding implies seeing, as we can pick $\mathbf{q} = \mathbf{0}$.) The number of packets seen by a node is precisely the rank of the node (see [8] for more details).

We introduce a new notion of packets that a node has "heard of". A node is said to have heard of a packet if it knows some linear combination involving that packet.

3. The new coding module

Our coding module works in the Galois field of size 3. At the beginning of every slot, the module has to decide what linear combination to transmit. Since there is full feedback, the module is fully aware of the current state of knowledge of each of the three receivers. Thus, it can compute the rank of each receiver. We denote the highest rank among the three receivers as m. Our coding module maintains an invariant that the transmission will never involve a packet whose index is greater than $(m+1)$. We denote the receiver(s) whose rank is m as the leader(s). We consider three cases:

3.1. All three receivers are leaders

In this case, all three receivers have a rank of m which means each has seen m packets. If $\mathbf{p_{m+1}}$ has not arrived yet, the module does nothing. Otherwise, since all transmissions have involved only the set of packets up to $\mathbf{p_{m+1}}$, there is exactly one unseen packet for each receiver within this set. This could be a different packet for each of the three receivers. The coding module selects a linear combination that if received successfully by any receiver, will reveal to that receiver its unseen packet, thereby guaranteeing innovation. This is done by simply forming a linear combination involving only the unseen packets of the three receivers. It can be verified that with a field of size 3, it is always possible to choose coefficients such that innovation is guaranteed for all three receivers.

3.2. There are two leaders

It can be shown by induction that at all times, at least one leader would have decoded all packets from 1 to m. Now, when there are two leaders, if exactly one leader has decoded all packets 1 to m, then the coding module performs the operations of case 3, treating this leader as the unique leader. If both leaders have decoded packets 1 to m, then module does the following.

If $\mathbf{p_{m+1}}$ has not arrived yet, the module transmits the oldest undecoded packet of the non-leader (if there are packets that the non-leader has heard of but not yet decoded, then they are preferred). Suppose $\mathbf{p_{m+1}}$ has arrived. Now, if it has already been decoded by the non-leader, then the module sends the sum of $\mathbf{p_{m+1}}$ and the oldest undecoded packet of the non-leader (again, if there are packets that the non-leader has heard of but not yet decoded, then they are preferred). Otherwise, the module sends $\mathbf{p_{m+1}}$ by itself.

3.3. Unique leader

In this case, the module computes the following sets for the two non-leaders:

H_1:= Set of packets heard of by first non-leader
H_2:= Set of packets heard of by second non-leader
D_1:= Set of packets decoded by first non-leader
D_2:= Set of packets decoded by second non-leader

Note that $D_1 \subseteq H_1$ and $D_2 \subseteq H_2$. We also define a universe set U consisting of packets $\mathbf{p_1}$ to $\mathbf{p_m}$, and also $\mathbf{p_{m+1}}$ if it has arrived. In this setting, the following sets partition the universe (refer to Figure 3):

- $S_1 = D_1 \cap D_2$
- $S_2 = D_1 \cap (H_2 \backslash D_2)$
- $S_3 = D_2 \cap (H_1 \backslash D_1)$
- $S_4 = (H_1 \backslash D_1) \cap (H_2 \backslash D_2)$
- $S_5 = D_1 \backslash H_2$
- $S_6 = D_2 \backslash H_1$
- $S_7 = (H_1 \backslash D_1) \backslash H_2$
- $S_8 = (H_2 \backslash D_2) \backslash H_1$
- $S_9 = U \backslash (H_1 \cup H_2)$

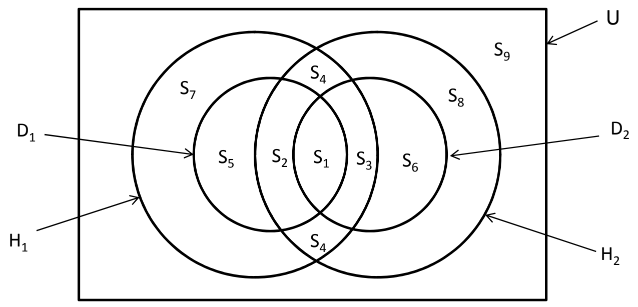

Figure 3: Sets used by the coding module

The coding module picks a linear combination depending on which of these sets $\mathbf{p_{m+1}}$ falls in, as follows:

Case 1 – $\mathbf{p_{m+1}}$ has not arrived: Check if S_4 is non-empty. If it is, then send the oldest packet in S_4. Otherwise, check if both S_2 and S_3 are non-empty. If they are, pick the oldest packet from each, and send their sum. If not, try the following pairs of sets: S_3 and S_5, else S_2 and S_6, else S_5 and S_6. If none of these pairs of sets work, then send the oldest packet in S_7 if it is non-empty. If not, try S_8, S_9, S_2, S_3, S_5 and S_6 in that order. If all of these are empty, then send nothing.

Case 2 – $\mathbf{p_{m+1}} \in S_1$: This is identical to case 1, except that $\mathbf{p_{m+1}}$ must also be added to the linear combination that case 1 suggests.

Case 3 – $\mathbf{p_{m+1}} \in S_2$: Send $\mathbf{p_{m+1}}$ added to another packet. The other packet is chosen to be the oldest packet in the first non-empty set in the following sets, tested in this particular order: S_3, S_4, S_6, S_8, S_7 and then S_9.

Case 4 – $\mathbf{p_{m+1}} \in S_3$: This is similar to the S_2 case (using symmetry) – test S_2, S_4, S_5, S_7, S_8 and then S_9.

Case 5 – $\mathbf{p_{m+1}} \in S_4$: Send $\mathbf{p_{m+1}}$ as it is.

Case 6 – $\mathbf{p_{m+1}} \in S_5$: Send $\mathbf{p_{m+1}}$ added to another packet. The other packet is chosen to be the oldest packet in the first non-empty set in the following sets, tested in the following order: S_3, S_6, S_4, S_8, S_7 and then S_9.

Case 7 – $\mathbf{p_{m+1}} \in S_6$: This is similar to the S_5 case (using symmetry) – test S_2, S_5, S_4, S_7, S_8 and then S_9.

Case 8 – $\mathbf{p_{m+1}} \in S_7$: Send $\mathbf{p_{m+1}}$ as it is.

Case 9 – $\mathbf{p_{m+1}} \in S_8$: Send $\mathbf{p_{m+1}}$ as it is.

Case 10 – $\mathbf{p_{m+1}} \in S_9$: Send $\mathbf{p_{m+1}}$ as it is.

In all these cases, the coefficient for the chosen packets must be selected to be either 1 or 2, in such a way that the resulting linear combination is innovative to any receiver that receives it, except if the receiver already knows all that the sender knows. It can be shown that such a choice is always possible with a field of size 3.

Remark 1 We conjecture based on the simulations that the algorithm maintains the following invariant – at most one of $H_1 \backslash D_1$ and $H_2 \backslash D_2$ is non-empty at any given time. If proved to be true, this observation can be used to simplify the algorithm.

4. The intuition behind the algorithm

The main idea behind the algorithm is to first of all guarantee innovation. It can be shown that the linear combination computed by this coding module is indeed innovative to any receiver that receives it. In addition to this requirement however, the module also tries to cause each receiver that has a successful reception to decode as many packets as possible.

An interesting property of this algorithm is that the transmitted linear combination always has at most two undecoded packets involved in it from any receiver's point of view. In other words, every transmission is essentially either an uncoded packet or the sum of two packets. This property leads to a nice structure in the knowledge space of the receivers, using which, we present a strategy to control the extent to which packets get mixed with each other, thereby controlling the decoding delay.

In order to explain this structure, we define the following relation. The ground set G of the relation contains all packets that have arrived at the sender so far, along with a fictitious all-zero packet that is known to all receivers even before transmission begins. The relation is defined with respect to a specific receiver. Two packets $\mathbf{p_x} \in G$ and $\mathbf{p_y} \in G$ are defined to be related to each other if the receiver knows at least one of $\mathbf{p_x} + \mathbf{p_y}$ and $\mathbf{p_x} + 2\mathbf{p_y}$.

Now, a packet added with two times the same packet gives $\mathbf{0}$ which is trivially known to the receiver. Hence, the relation is reflexive. It is also symmetric since addition is a commutative operation. Now, for any $\mathbf{p_x}, \mathbf{p_y}, \mathbf{p_z}$ in G, if a receiver knows $\mathbf{p_x} + \alpha\mathbf{p_y}$ and $\mathbf{p_y} + \beta\mathbf{p_z}$, then it can compute either $\mathbf{p_x} + \mathbf{p_z}$ or $\mathbf{p_x} + 2\mathbf{p_z}$ by canceling out the $\mathbf{p_y}$, for $\alpha = 1$ or 2 and $\beta = 1$ or 2. Therefore the relation is also transitive and is thus an equivalence relation. It defines a partition on the ground set, namely the equivalence classes, which provide a structured way to represent the knowledge of the node. It can be seen that the class containing the all-zero packet is precisely the set of decoded packets (D_1 or D_2). Packets that have not been involved in any of the successfully received linear combinations so far will form singleton equivalence classes. These correspond to the packets that the receiver has not heard of ($U \backslash H_1$ or $U \backslash H_2$).

We say a class is nontrivial if it is neither a singleton class nor the class of decoded packets. Thus, nontrivial classes contain the packets that have been heard of but not decoded. Revealing any packet in a class will reveal the entire class to the node. The number of nontrivial classes is thus the number of packets that the node needs to know in order to decode all packets it has heard of. This number is thus a measure of how far away a node is from decoding all packets it has heard of. We call this number the *deficit of the node.*

For instance, revealing a packet from $H_1 \backslash D_1$ will allow the entire class containing that packet to be decoded by receiver 1. The algorithm ensures that a packet from $H_1 \backslash D_1$

or $H_2 \backslash D_2$ is revealed whenever possible, as opposed to a packet that the receiver has not heard of. This ensures that the deficit is reduced whenever possible. As a result, the deficit drops to zero frequently, thereby causing the node to decode packets.

5. Performance of the algorithm

5.1. Throughput optimality

The algorithm has been designed in such a way that innovation is guaranteed to all the receivers whenever possible. Packet $\mathbf{p_{m+1}}$ is always included in the linear combination if it has arrived, in order to guarantee innovation to the leader. If both the other receivers have also not decoded it, then sending $\mathbf{p_{m+1}}$ by itself satisfies the innovation guarantee. This happens in cases 5, 8, 9 and 10.

If however, some receiver has already decoded it as in the other cases, then another packet is included in the linear combination that the receiver has not yet decoded, thereby ensuring innovation. While choosing such a packet, preference is given to packets that the receiver has heard of, as revealing such a packet will cause several packets to be decoded at once.

If $\mathbf{p_{m+1}}$ has not yet arrived, then the leader is already satisfied. For the other two receivers, the transmission is selected in such a way that it simultaneously reveals an undecoded packet to both of them whenever possible.

We can show that in all these cases, over a field of size 3, the coefficients can also be chosen carefully to guarantee innovation for all those who receive the linear combination successfully. This discussion is summarized in the following theorem.

Theorem 1 *The coding module satisfies the innovation guarantee property.*

This means that the algorithm achieves optimal throughput, *i.e.*, for all $\rho < 1$, the decoding delay and the queue at the sender will remain stable.

5.2. Decoding delay

We now study the delay experienced by an arbitrary arrival before it gets decoded by one of the receivers, say receiver 1. We consider a system where μ is fixed at 0.5. The value of ρ is varied as follows: 0.95, 0.97, 0.98, 0.99 and 0.995. We plot the expected decoding delay per packet averaged across the three receivers, as a function of $\left(\frac{1}{1-\rho}\right)$ in Figure 4. We also plot the log of the same quantities in Figure 5. The value of the delay is averaged over 500000 time slots for the first three points and 10^6 time slots for the last two points.

Figure 4 shows that the growth of the expected decoding delay is linear in $\left(\frac{1}{1-\rho}\right)$ as ρ approaches 1. Figure 5 confirms this behavior – we can see that the slope on the plot of the logarithm of these quantities is indeed close to 1. This observation leads to the following conjecture:

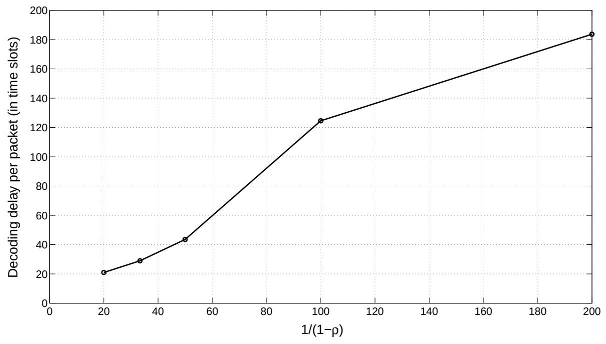

Figure 4: Decoding delay as load approaches capacity

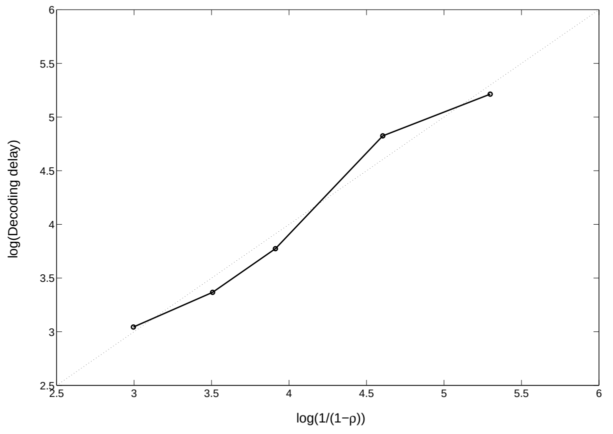

Figure 5: Decoding delay – log-log plot

Conjecture 1 *For the newly proposed coding module, the expected decoding delay per packet from one particular receiver's point of view grows as* $O\left(\frac{1}{1-\rho}\right)$, *which is asymptotically optimal.*

5.3. Queue size

The queue update rule is as follows – the sender drops a packet if all the receivers have decoded it. This means that by Little's law, the expected queue size will be proportional to the time a packet spends in the system before it is decoded. Thus, if the expected decoding delay is indeed $O\left(\frac{1}{1-\rho}\right)$ as conjectured above, then the drop-when-decoded queue update rule will ensure that the expected queue size at the sender is $O\left(\frac{1}{1-\rho}\right)$, which is asymptotically optimal.

6. Conclusions

For a three receiver packet erasure broadcast channel with feedback, we have proposed a new coding scheme that makes use of the feedback to dynamically adapt the code. As argued earlier, $\Theta\left(\frac{1}{1-\rho}\right)$ is an asymptotic lower bound on the decoding delay. We have observed through simulations that this lower bound seems to be achieved by our scheme, which would imply the asymptotic optimality of our coding module in terms of decoding delay. We conjecture that this is indeed true. All these delay benefits are obtained without compromising on throughput. If the conjecture is true, then the expected queue size of undecoded packets is also $O\left(\frac{1}{1-\rho}\right)$, which is asymptotically optimal. Thus, our scheme also simplifies the queue management at the sender. In the future, we wish to extend this approach to an arbitrary number of receivers. Also, we wish to make the algorithm robust to delays and erasures in the feedback.

References

[1] D. S. Lun, M. Médard, and M. Effros, "On coding for reliable communication over packet networks," in *42nd Annual Allerton Conference on Communication, Control, and Computing*, September – October 2004.

[2] M. Luby, "LT codes," in *Proc. of 43rd Annual IEEE Symposium on Foundations of Computer Science*, November 2002, pp. 271–282.

[3] A. Shokrollahi, "Raptor codes," in *Proc. of 2004 IEEE International Symposium on Information Theory (ISIT)*, July 2004.

[4] E. Martinian, "Dynamic information and constraints in source and channel coding," PhD Thesis, Massachusetts Institute of Technology, Dept. of EECS, Sep. 2004.

[5] S. Sanghavi, "Intermediate performance of rateless codes," in *Proc. of 2007 IEEE Information Theory Workshop*, September 2007.

[6] A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," in *Proc. of 2004 IEEE Information Theory Workshop*, October 2004.

[7] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *Proc. of NetCod*, 2008.

[8] J. K. Sundararajan, D. Shah, and M. Médard, "ARQ for network coding," in *Proc. of 2008 IEEE International Symposium on Information Theory (ISIT)*, 2008.

[9] T. Tran, T. Nguyen, and B. Bose, "A joint network-channel coding technique for single-hop wireless networks," in *Proc. of NetCod*, jan 2008.

[10] M. Durvy, C. Fragouli, and P. Thiran, "Towards reliable broadcasting using ACKs," in *Proc. of 2007 IEEE International Symposium on Information Theory (ISIT)*, 2007.

[11] C. Fragouli, D. S. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. of 2007 Conference on Information Sciences and Systems (CISS)*, March 2007.

[12] J. J. Hunter, *Mathematical Techniques of Applied Probability, Vol. 2, Discrete Time Models: Techniques and Applications.* NY: Academic Press, 1983.