Feedback-Based Online Network Coding

Jay Kumar Sundararajan, *Member, IEEE*, Devavrat Shah, *Senior Member, IEEE*, Muriel Médard, *Fellow, IEEE*, and Parastoo Sadeghi, *Senior Member, IEEE*

***Abstract*— Current approaches to the practical implementation of network coding are batch-based, and often do not use feedback, except possibly to signal completion of a file download. In this paper, the various benefits of using feedback in a network coded system are studied. It is shown that network coding can be performed in a completely online manner, without the need for batches or generations, and that such online operation does not affect the throughput. Although these ideas are presented in a single-hop packet erasure broadcast setting, they naturally extend to more general lossy networks, which employ network coding in the presence of feedback. The impact of feedback on sender-side queue size and receiver-side decoding delay is studied in an asymptotic sense as the traffic load approaches capacity. Different notions of decoding delay are considered, including an order-sensitive notion, which assumes that packets are useful only when delivered in order. Strategies for adaptive coding based on feedback are presented. Our scheme achieves throughput optimality and asymptotically optimal sender queue size and is conjectured to achieve asymptotically optimal in-order delivery delay for any number of receivers. This paper may be viewed as a natural extension of Automatic Repeat reQuest to coded networks.**

***Index Terms*— ARQ, decoding delay, network coding.**

I. Introduction

THIS paper is a step towards low-delay, high-throughput solutions based on network coding, for real-time data streaming applications over a packet erasure network. In particular, it considers the role of feedback for queue management and delay control in such systems.

A. Background

Reliable communication over a network of packet erasure channels is a well studied problem, especially when there is no feedback. We survey some of the approaches from the literature.

1) Digital Fountain Codes: The digital fountain codes ([1], [2]) constitute a well-known approach to this problem. From a block of k transmit packets, the sender generates random linear combinations in such a way that the receiver can, with high probability, decode the block once it receives *any* set of slightly more than k linear combinations. This approach has low complexity and requires no feedback, except to signal successful decoding of the block. However, fountain codes are designed for a point-to-point erasure channel and in their original form, do not extend readily to a network setting. In a two-link tandem network, an end-to-end fountain code with simple forwarding at the middle node will result in throughput loss. If the middle node chooses to decode and re-encode an entire block, the scheme will be sub-optimal in terms of delay, as pointed out by [3]. In this sense, the fountain code approach is not composable across links. For the special case of tree networks, there has been some recent work on composing fountain codes across links by enabling the middle node to re-encode even before decoding the entire block [4].

2) Random Linear Network Coding: Network coding was originally introduced for error-free networks with specified link capacities ([5], [6]), and was extended to erasure networks [7]. The random linear network coding solution of [8] does not require decoding at intermediate nodes and can be applied in any network. Each node transmits a random linear combination of all coded packets it has received so far. With high probability, the transmitted packet will have the *innovation guarantee property*, *i.e.*, it will be *innovative*[1] to every receiver that receives it successfully, except if the receiver already knows as much as the sender. In [8], this scheme is shown to achieve capacity for a multicast session.

Although fountain codes and random linear network coding are both rateless, the encoding operation is performed on a block (or generation) of packets. There is no guarantee that the receiver will be able to extract any of the original packets from the coded packets until the entire block has been decoded, resulting in a decoding delay.

This is not a problem if the higher layers will anyway use a block only as a whole (*e.g.*, file download). This corresponds to traditional approaches in information theory where the message is assumed to be useful only as a whole. No incentive is placed on decoding "a part of the message" using a part of the codeword. However, many applications today involve broadcasting a continuous stream of packets in real-time (*e.g.*, video streaming). Sources generate a stream of messages which have an intrinsic temporal ordering. Playback is possible only till the point up to which all packets have been recovered, which we call *the front of contiguous knowledge*. There is incentive to decode the older messages earlier, as this

Manuscript received April 10, 2009; revised December 8, 2015; accepted March 31, 2017. Date of publication May 31, 2017; date of current version September 13, 2017. This paper was presented in part at the 2007 IEEE ITW, the 2008 IEEE ISIT, the 2008 ISITA, and the 2009 IEEE NetCod.

J. K. Sundararajan was with the Massachusetts Institute of Technology, Cambridge, MA 02139 USA. He is now with Qualcomm Research, San Diego, CA 92121 USA (e-mail: sjaykumar@gmail.com).

D. Shah and M. Médard are with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology, Cambridge, MA 02139 USA (e-mail: devavrat@mit.edu; medard@mit.edu).

P. Sadeghi is with The Australian National University, Canberra, ACT 2601, Australia (e-mail: parastoo.sadeghi@anu.edu.au).

Communicated by R. Berry, Associate Editor for Communication Networks.

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIT.2017.2710192

[1]An innovative packet is a linear combination of packets which is linearly independent of previously received linear combinations, and thus conveys new information.

will reduce the playback latency. In block-based approaches, block sizes will have to be large to ensure high throughput but if playback can begin only after receiving a full block, then large blocks will imply a large delay.

This raises an interesting question: can we code in such a way that playback can begin even before the full block is received? In other words, we are more interested in packet delay than block delay. These issues have been studied by [9], [10], and [11] in a point-to-point setting. However, in a network setting, the problem is not well understood. Moreover, these works do not consider the queue management aspects of the problem. In related work, [12], [13] address the question of how many original packets are revealed before the whole block is decoded in a fountain code setting. However, playback performance may depend on not only *how much data* reaches the receiver in a given time, but also *which part of the data*, *i.e.*, the order in which they are recovered.

3) Priority Encoding Transmission: Priority encoding transmission (PET) ([14]), addresses this problem with a code for the erasure channel that ensures that the first (or highest priority) i messages can be decoded using the first k_i coded packets, where k_i increases with decreasing priority. In [15] and [16], this is extended to systems that perform network coding. A concatenated network coding scheme is proposed in [16], with a delay-mitigating pre-coding stage. This scheme guarantees that the kth innovative reception will enable the receiver to decode the kth message. In such schemes however, the ability to decode messages in order requires a reduction in throughput because of the pre-coding stage.

B. Motivation

The main motivation for our current work is that the availability of feedback brings the hope of simultaneously achieving the best possible throughput along with minimal packet delay and queue size.

Reliable communication over a point-to-point packet erasure channel with full feedback can be achieved using the Automatic Repeat reQuest (ARQ) scheme – whenever a packet gets erased, the sender retransmits it. Every successful reception conveys a new packet, implying throughput optimality. Moreover, this new packet is always the next unknown packet, which implies the lowest possible packet delay. Since there is feedback, the sender never stores anything the receiver already knows, implying optimal queue size. Thus, this simple scheme simultaneously achieves the optimal throughput, delay and queue size. Moreover, the scheme is completely online and not block-based.

However, ARQ is not sufficient beyond a single point-to-point link. Coding across packets is necessary to achieve optimal throughput, even if we allow feedback. For instance, in the network coding context, link-by-link ARQ cannot achieve the multicast capacity of the *butterfly network* from network coding literature [5]. Similarly, ARQ is sub-optimal for broadcast-mode links because retransmitting a packet that some receivers did not get is wasteful for the others that already have it. In contrast, network coding achieves the multicast capacity of any network and also readily extends to networks with broadcast-mode links. Thus, in such situations, coding is indispensable from a throughput perspective.

This leads to the question – how to combine the benefits of ARQ and network coding? The goal is to extend ARQ's desirable properties in the point-to-point context to systems that require coding across packets.

The problem with applying ARQ to a coded system is that a new reception may not always reveal the next unknown packet to the receiver, but may bring in a linear equation involving the packets. In conventional ARQ, upon receiving an ACK, the sender drops the ACKed packet and transmits the next one. But in a coded system, upon receiving an ACK for a linear equation, it is not clear which linear combination the sender should pick for its next transmission to obtain the best system performance. This is important because, if the receiver has to collect many equations before it can decode the unknowns involved, this could lead to a large decoding delay.

A related question is: upon receiving the ACK for a linear equation, which packet can be excluded from future coding, *i.e.*, which packet can be dropped from the sender's queue? If packets arrive at the sender according to some stochastic process, (as in [17] and [18]) and links are lossy (as in [7] and [8]), then the queue management aspect of the problem also becomes important.

Dropping packets that all receivers have decoded would clearly not affect reliability. However, storing all undecoded packets may be suboptimal. Consider a situation where the sender has n packets $\mathbf{p_1}, \mathbf{p_2} \ldots, \mathbf{p_n}$, and all receivers have received $(n-1)$ linear combinations: $(\mathbf{p_1}+\mathbf{p_2})$, $(\mathbf{p_2}+\mathbf{p_3})$, $\ldots$, $(\mathbf{p_{n-1}}+\mathbf{p_n})$. A drop-when-decoded scheme will not allow the sender to drop any packet, since no packet can be decoded by any receiver yet. However, the backlog in the amount of information, also called the *virtual queue* ([17], [18]), has a size of just 1. We ideally want the physical queue to track the virtual queue in size. In this example, it would be sufficient if the sender stores any one $\mathbf{p_i}$ in order to ensure reliable delivery.

If we have feedback in a system with network coding, what is the best possible tradeoff between throughput, delay and queue size? In particular, how close can we get to the performance of ARQ for the point-to-point case? These are the questions we address in this paper.

II. Our Contribution

We show that with the proper use of feedback, it is possible to perform network coding in a completely online manner similar to ARQ schemes, without the need for a block-based approach. We study the benefits of feedback in a coded network in terms of the following two aspects – queue management and decoding delay.

We consider a system with stochastic packet arrivals and study the problem using a queuing theory approach. Let λ be the arrival rate, μ be the channel quality parameter, and $\rho \triangleq \lambda/\mu$ be the load factor. We consider asymptotics when the ρ approaches 1, while keeping either λ or μ fixed at a value less than 1. The optimal throughput requirement means that the queue of undelivered packets is stable for all $\rho < 1$. We treat packets as vectors over a finite field. We restrict

TABLE I
EXAMPLE OF THE DROP-WHEN-SEEN ALGORITHM

Time	Sender's queue	Transmitted packet	Channel state	A		B	
				Decoded	Seen but not decoded	Decoded	Seen but not decoded
1	$\mathbf{p_1}$	$\mathbf{p_1}$	$\rightarrow$ A, $\nrightarrow$ B	$\mathbf{p_1}$	-	-	-
2	$\mathbf{p_1}, \mathbf{p_2}$	$\mathbf{p_1} \oplus \mathbf{p_2}$	$\rightarrow$ A, $\rightarrow$ B	$\mathbf{p_1}, \mathbf{p_2}$	-	-	$\mathbf{p_1}$
3	$\mathbf{p_2}, \mathbf{p_3}$	$\mathbf{p_2} \oplus \mathbf{p_3}$	$\nrightarrow$ A, $\rightarrow$ B	$\mathbf{p_1}, \mathbf{p_2}$	-	-	$\mathbf{p_1}, \mathbf{p_2}$
4	$\mathbf{p_3}$	$\mathbf{p_3}$	$\nrightarrow$ A, $\rightarrow$ B	$\mathbf{p_1}, \mathbf{p_2}$	-	$\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$	-
5	$\mathbf{p_3}, \mathbf{p_4}$	$\mathbf{p_3} \oplus \mathbf{p_4}$	$\rightarrow$ A, $\nrightarrow$ B	$\mathbf{p_1}, \mathbf{p_2}$	$\mathbf{p_3}$	$\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}$	-
6	$\mathbf{p_4}$	$\mathbf{p_4}$	$\rightarrow$ A, $\rightarrow$ B	$\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}$	-	$\mathbf{p_1}, \mathbf{p_2}, \mathbf{p_3}, \mathbf{p_4}$	-

our attention to linear network coding. Therefore, the state of knowledge of a node can be viewed as a vector space over the field (see Section III for further details).

A. Queue Management

We propose a new acknowledgment mechanism that *acknowledges degrees of freedom*[2] *instead of original decoded packets*. Based on this new form of ACKs, we propose an online coding module that naturally generalizes ARQ to coded systems. The code implies a queue update algorithm that ensures that *the physical queue size at the sender will track the backlog in degrees of freedom.*

The key intuition is that *the sender need not use for coding (and hence need not store) any information that has already been received by all the receivers.* To be specific, we can ensure reliability even if we restrict the sender's transmit packet to be chosen from a subspace that is independent[3] of the subspace representing the common knowledge available at all the receivers. Therefore, the queue simply needs to store a basis for a coset space with respect to the subspace of knowledge common to all the receivers. We define a specific way of computing this basis using the new notion of a node "seeing" a message packet, which is defined below.

Definition 1 (Index of a Packet): For any positive integer k, the kth packet that arrives at the sender is said to have an *index* k.

Definition 2 (Seeing a Packet): 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 $\mathbf{q} = \mathbf{0}$.)

In conventional ARQ, a receiver ACKs a packet upon decoding it successfully. However, in our scheme **a receiver ACKs a packet when it sees the packet**. Our new scheme is called the *drop-when-seen* algorithm because the sender *drops a packet if all receivers have seen (ACKed) it.*

Since decoding implies seeing, the sender's queue is expected to be shorter under our scheme compared to the drop-when-decoded scheme. However, we will need to show that in spite of dropping seen packets even before they are decoded, we can still ensure reliable delivery. To prove this, we present a deterministic coding scheme that uses only unseen packets and still guarantees that the coded packet will **simultaneously cause each receiver that receives it successfully, to see its next unseen packet**. We will prove later that seeing a new packet translates to receiving a new degree of freedom. This means, the innovation guarantee property is satisfied and therefore, reliability and 100% throughput can be achieved (see Algorithm 2 (b) and corresponding Theorems 6 and 8 in Section IV-C).

The intuition is that if all receivers have seen $\mathbf{p}$, then their uncertainty can be resolved using only packets with index more than that of $\mathbf{p}$ because after decoding these packets, the receivers can compute $\mathbf{q}$ and hence obtain $\mathbf{p}$ as well. Therefore, even if the receivers have not decoded $\mathbf{p}$, no information is lost by dropping it, provided it has been seen by all receivers.

Next, we present an example that explains our algorithm for a simple two-receiver case. Section IV-C3 extends this scheme to more receivers.

Example: Table I shows a sample of how the proposed idea works in a packet erasure broadcast channel with two receivers A and B. The sender's queue is shown after the arrival point and before the transmission point of a slot (see Section III for details on the setup). In each slot, based on the ACKs, the sender identifies the next unseen packet for A and B. If they are the same packet, then that packet is sent. If not, their XOR is sent. It can be verified that with this rule, every reception causes each receiver to see its next unseen packet.

In slot 1, $\mathbf{p_1}$ reaches A but not B. In slot 2, $(\mathbf{p_1} \oplus \mathbf{p_2})$ reaches A and B. Since A knows $\mathbf{p_1}$, it can also decode $\mathbf{p_2}$. As for B, it has now seen (but not decoded) $\mathbf{p_1}$. At this point, since A and B have seen $\mathbf{p_1}$, the sender drops it. This is fine even though B has not yet decoded $\mathbf{p_1}$, because B will eventually decode $\mathbf{p_2}$ (in slot 4), at which time it can obtain $\mathbf{p_1}$. Similarly, $\mathbf{p_2}$, $\mathbf{p_3}$ and $\mathbf{p_4}$ will be dropped in slots 3, 5 and 6 respectively. However, the drop-when-decoded policy will drop $\mathbf{p_1}$ and $\mathbf{p_2}$ in slot 4, and $\mathbf{p_3}$ and $\mathbf{p_4}$ in slot 6. Thus, our new strategy clearly keeps the queue shorter. This is formally proved in Theorem 1 and Theorem 6. The example also shows that it is fine to drop packets before they are decoded. Eventually, the future packets will arrive, thereby allowing the decoding of all the packets.

Implications of Our New Scheme:

- **Queue size:** The physical queue size is upper-bounded by the sum of the backlogs in degrees of freedom between the sender and all the receivers. This fact implies

[2]Here, *degree of freedom* refers to a new dimension in the vector space representing the sender's knowledge.

[3]A subspace S_1 is said to be *independent* of another subspace S_2 if $S_1 \cap S_2 = \{\mathbf{0}\}$. See [19] for more details.

that as the traffic load approaches capacity (as load factor $\rho \to 1$), the expected size of the physical queue at the sender is $O\left(\frac{1}{1-\rho}\right)$. This is the same order as for single-receiver ARQ, and hence, is order-optimal.

- **Queuing analysis:** Our scheme forms a natural bridge between the virtual and physical queue sizes. It can be used to extend results on the stability of virtual queues such as [17], [18], and [20] to physical queues. Moreover, various results obtained for virtual queues from traditional queuing theory, such as the transform based analysis for the queue size of M/G/1 queues, or even a Jackson network type of result [8], can be extended to the physical queue size of nodes in a network coded system.
- **Simple queue management:** Our approach based on *seen packets* ensures that the sender does not have to store linear combinations of the packets in the queue to represent the basis of the coset space. Instead, it can store the basis using the original uncoded packets themselves. Therefore, the queue follows a simple first-in-first-out service discipline.
- **Online encoding:** All receivers see packets in the same order in which they arrived at the sender. This gives a guarantee that the information deficit at the receiver is restricted to a set of packets that advances in a streaming manner and has a stable size (namely, the set of unseen packets). In this sense, the proposed encoding scheme is truly online.
- **Easy decoding:** Every transmitted linear combination is sparse – at most n packets are coded together for the n receiver case. This reduces the decoding complexity as well as the overhead for embedding the coding coefficients in the packet header.
- **Extensions:** We present our scheme for a single packet erasure broadcast channel. However, our algorithm is composable across links and can be applied to a tandem network of broadcast links. With suitable modifications, it can potentially be applied to a more general setup like the one in [7] provided we have feedback. Such extensions are discussed further in Section VII.

B. Decoding Delay

The drop-when-seen algorithm and the associated coding module do not guarantee that the seen packets will be decoded immediately. In general, there will be a delay in decoding, as the receiver will have to collect enough linear combinations involving the unknown packets before being able to decode the packets.

Online feedback-based adaptation of the code with the goal of minimizing decoding delay has been studied in the literature and related earlier work is discussed in II-C.

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. We introduce a new requirement that the growth of the average decoding delay as a function of $\frac{1}{1-\rho}$ as $\rho \to 1$ should meet this lower bound asymptotically, *i.e.*, we aim to guarantee asymptotically optimal decoding delay as the system load approaches capacity. 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. We also study a stronger notion of delay, namely the *delivery delay*, which measures delay until the point when the packet can be delivered to the application above, with an in-order delivery constraint.

We provide a new coding module for any number of receivers that **achieves optimal throughput and asymptotically optimal queue size at the sender. We conjecture that it also achieves an asymptotically optimal decoding and delivery delay as the system approaches capacity, 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 $\Theta\left(\frac{1}{1-\rho}\right)$. With more than one receiver, 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 these conjectures through simulations for values of ρ that are very close to 1. It is useful to note that asymptotically optimal decoding delay translates to asymptotically optimal expected queue occupancy at the sender using the simple queuing rule of dropping packets that have been decoded by all receivers. Note that the conjectures, if true, would also imply that the memory requirement on the receiver side for decoding and resequencing would also grow as $O\left(\frac{1}{1-\rho}\right)$ by Little's theorem.

Adaptive coding allows the sender's code to incorporate receivers' states of knowledge and thereby enables the sender to control the evolution of the front of contiguous knowledge. Our schemes may thus be viewed as a step towards feedback-based control of the tradeoff between throughput and decoding delay, along the lines suggested in [21].

C. Related Earlier Work

In [22], Shrader and Ephremides study the queue stability and delay of block-based random linear coding versus uncoded ARQ for stochastic arrivals in a broadcast setting. However, this work does not consider the combination of coding and feedback in one scheme. In related work, [23] studies load-dependent variable sized coding blocks with ACKs at the end of a block, using a bulk-service queue model. The main difference in our work is that receivers ACK packets even before decoding them, and this enables the sender to perform online coding.

Sagduyu and Ephremides [24] consider online feedback-based adaptation of the code, and propose a coding scheme for the case of two receivers. This work focuses on the maximum possible stable throughput, and does not consider the use of feedback to minimize queue size or decoding delay. In [25], the authors study the throughput of a block-based coding scheme, where receivers acknowledge the successful decoding of an entire block, allowing the sender to move to the next block. Next, they consider the option of adapting the code based on feedback for the multiple receiver case. They build on the two-receiver case of [24] and propose a greedy deterministic coding scheme that may not be throughput optimal, but picks a linear combination such that the number of receivers that immediately decode a packet is maximized. In contrast, in our work we consider throughput-optimal policies that aim to minimize queue size and delay.

In [26], Lacan and Lochin propose an erasure coding algorithm called Tetrys to ensure reliability in spite of losses on the acknowledgment path. While this scheme also employs coding in the presence of feedback, their approach is to make minimal use of the feedback, in order to be robust to feedback losses. As opposed to such an approach, we investigate how best to use the available feedback to improve the coding scheme and other performance metrics. For instance, in the scheme in [26], packets are acknowledged (if at all) only when they are decoded, and these are then dropped from the coding window. However, we show in this work that by dropping packets when they are seen, we can maintain a smaller coding window without compromising on reliability and throughput. A smaller coding window translates to lower encoding complexity and smaller queue size at the sender.

The use of ACKs and coded retransmissions in a packet erasure broadcast channel has been considered for multiple unicasts [27] and multicast ([28]–[31]). The main goal of these works however, is to optimize the throughput. Other metrics such as queue management and decoding delay are not considered. In our work, we focus on using feedback to optimize these metrics as well, in addition to achieving 100% throughput in a multicast setting. Our coding module (in Section IV-C5) is closely related to the one proposed by Larsson in an independent work [30]. However, our algorithm is specified using the more general framework of seen packets, which allows us to derive the drop-when-seen queue management algorithm and bring out the connection between the physical queue and virtual queue sizes. Reference [30] does not consider the queue management problem. Moreover, using the notion of seen packets allows our algorithm to be compatible even with random coding. This in turn enables a simple ACK format and makes it suitable for practical implementation. (See Remark 2 for further discussion.)

Coding for per-packet delay has been studied in earlier work by Martinian *et al.* [9]. However, that work considered a point-to-point setting unlike our broadcast scenario. The problem of the delay for recovering packets from a file has been studied in the rateless code framework with or without feedback, by [12] and [13]. Reference [10] also considered the problem of coding for delay using feedback. The setting there is in terms of a fixed delay model for point-to-point communication, where each packet has a deadline by which it needs to be delivered. A packet which does not meet its deadline is considered to be in error, and the corresponding error exponents are characterized. In contrast, we consider the expected per-packet delay in a queuing theoretic framework, with no strict deadlines. Besides, our setting is a point-to-multipoint (broadcast) packet erasure channel.

For a packet erasure broadcast channel with two receivers, Durvy *et al.* [32] have proposed a feedback-based throughput optimal coding scheme that ensures that every successful innovative reception at any receiver will cause it to decode an original packet. This property is called *instantaneous decodability*. Keller *et al.* [33] also studied this problem and proposed and compared several algorithms to reduce the decoding delay. This work did not consider the in-order delivery problem. With prior knowledge of the erasure pattern, [33] gave an offline algorithm that achieves optimal delay and throughput for the case of three receivers. However, in the online case, even with only three receivers, [32] shows through an example (Example V.1) that it is not possible to simultaneously ensure instantaneous decodability as well as throughput optimality.

Both [32] and [33] consider the transmission of a given finite set of packets. In contrast, [24] assumes that packets arrive at the source according to a stochastic process in a streaming manner and proposes a coding scheme for two receivers. The focus however, is to ensure stable throughput and not low delay. In [25], the authors propose a greedy coding scheme for the case of more than 2 receivers, which aims to maximize the number of receivers that can decode a packet instantaneously, at the expense of losing throughput.

Reference [34] considers heterogeneous channels to the receivers, and proposes a systematic online encoding scheme that sends uncoded packets to enable frequent decoding at the receivers. However, no characterization of the asymptotic behavior of the decoding or delivery delay is provided.

In our current work, we focus on throughput-optimal schemes in a stochastic arrival setting. In the light of earlier results that instantaneously decodability and throughput optimality cannot be simultaneously guaranteed with more than 2 receivers, we aim for a relaxed requirement of asymptotically optimal delay where the asymptotics are in the limit of the load factor $\rho \to 1$. However, while the earlier work did not consider the in-order delivery constraint, we study both decoding and delivery delay. For the case of two receivers, it can be shown that the scheme in [32] satisfies this requirement due to the instantaneous decodability property, *i.e.*, the scheme achieves the asymptotically optimal average decoding delay, and seems to achieve it for delivery delay as well based on simulations. In fact, the algorithm of [32] is a special case of our algorithm.

Our work provides a new coding module that achieves throughput optimality and is conjectured to achieve asymptotically optimal decoding and delivery delay for any number of receivers.

D. Organization

The rest of the paper is organized as follows. Section III describes the packet erasure broadcast setting. Section IV is concerned with adaptive codes that minimize the sender's queue size. In Section IV-A, we define and analyze a baseline algorithm that drops packets only when they have been decoded by all receivers. Section IV-B presents a generic form of our newly proposed algorithm, and introduces the idea of excluding from the sender's queue, any knowledge that is common to all receivers. We show that the algorithm guarantees that the physical queue size tracks the virtual queue size. Section IV-C presents an easily implementable variant of the generic algorithm of Section IV-B, called the drop-when-seen algorithm. The drop-when-seen algorithm consists of a queuing module that provides guarantees on the queue size, and a coding module that provides guarantees on reliability and throughput, while complying with the queuing module. In Section VI, we investigate adaptive codes aimed at minimizing the receivers' decoding delay. We propose a new coding module that is proved to be throughput optimal and conjectured to be asymptotically optimal in terms of delay. Section VII presents some ideas on extending the algorithms to more general topologies and scenarios. Finally, Section VIII gives the conclusions.

III. The Setup

In this paper, we consider a communication problem where a sender wants to broadcast a stream of data to n receivers. The data are organized into *packets*, which are essentially vectors of fixed size over a finite field $\mathbb{F}_q$. A packet erasure broadcast channel connects the sender to the receivers. Time is slotted. The details of the queuing model and its dynamics are described next.

A. The Queuing Model

The sender is assumed to have an infinite buffer, *i.e.*, a queue with no preset size constraints. We assume that the sender is restricted to use linear codes. Thus, every transmission is a linear combination of packets from the incoming stream that are currently in the buffer. The vector of coefficients used in the linear combination summarizes the relation between the coded packet and the original stream. We assume that this coefficient vector is embedded in the packet header. A node can compute any linear combination whose coefficient vector is in the linear span of the coefficient vectors of previously received coded packets. In this context, the state of knowledge of a node can be defined as follows.

Definition 3 (Knowledge of a Node): The *knowledge of a node* at some point in time is the set of all linear combinations of the original packets that the node can compute, based on the information it has received up to that point. The coefficient vectors of these linear combinations form a vector space called the *knowledge space* of the node.

In the following definitions, V denotes the sender's knowledge space, and V_j denote the knowledge space of receiver j for $j = 1, 2, \ldots, n$.

Definition 4 (Virtual Queue): For $j = 1, 2, \ldots, n$, the size of the jth virtual queue corresponding to receiver j is defined to be the difference between the dimension of the V and that of V_j.

Definition 5 (Degree of Freedom): The term *degree of freedom* refers to one dimension in the knowledge space of a node. It corresponds to one packet worth of data.

Definition 6 (Innovative Packet): A coded packet with coefficient vector $\mathbf{c}$ is said to be *innovative* to a receiver j if $\mathbf{c} \notin V_j$. Such a packet, if successfully received, will increase the dimension of the receiver's knowledge space by one unit.

Definition 7 (Innovation Guarantee Property): A coding scheme is said to have the *innovation guarantee property* if in every slot, the coefficient vector of the transmitted linear combination is in $V \backslash V_j$ for every j such that $V_j \neq V$. In other words, the transmission is innovative to every receiver except when the receiver already knows everything that the sender knows.

We use the notion of a virtual queue to represent the backlog between the sender and receiver in terms of linear degrees of freedom. This notion was also used in [17], [18], and [20]. There is one virtual queue for each receiver. We will use the term *physical queue* to refer to the sender's actual buffer, in order to distinguish it from the virtual queues. Note that the virtual queues do not correspond to real storage.

B. Arrivals

Packets arrive into the sender's physical queue according to a Bernoulli process[4] of rate λ. An arrival at the physical queue translates to an arrival at each virtual queue since the new packet is a new degree of freedom that the sender knows, but none of the receivers knows.

C. Service

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. The receivers are assumed to be capable of detecting an erasure.

We only consider coding schemes that satisfy the innovation guarantee property. This property implies that if the virtual queue of a receiver is not empty, then a successful reception reveals a previously unknown degree of freedom to the receiver and the virtual queue size decreases by one unit. We can thus map a successful reception by some receiver to one unit of service of the corresponding virtual queue. This means, in every slot, each virtual queue is served independently of the others with probability μ.

The relation between the service of the virtual queues and the service of the physical queue depends on the queue update scheme used, and will be discussed separately under each update policy.

D. Feedback

We assume perfect delay-free feedback. In Algorithm 1 below, feedback is used to indicate successful decoding. For all the other algorithms, the feedback is needed in every slot to indicate the occurrence of an erasure.

E. Timing

Figure 1 shows the relative timing of events within a slot. All arrivals are assumed to occur *just after the beginning* of the slot. The point of transmission is after the arrival point. For simplicity, we assume very small propagation time. Specifically, we assume that the transmission, unless erased by the channel, reaches the receivers before they send feedback for that slot and feedback from all receivers reaches the sender *before the end of the same slot*. Thus, the feedback

[4]We have assumed Bernoulli arrivals for ease of exposition. However, we expect the results to hold for more general arrival processes as well.

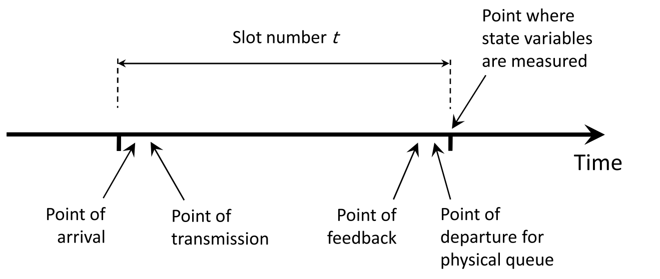
Slot number t
Point where state variables are measured
Time
Point of arrival
Point of transmission
Point of feedback
Point of departure for physical queue

Fig. 1. Relative timing of arrival, service and departure points within a slot.

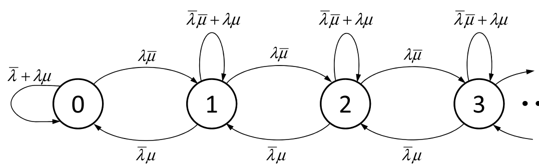
$\bar{\lambda} + \lambda\mu$
$\lambda\bar{\mu}$
$\bar{\lambda}\bar{\mu} + \lambda\mu$
$\bar{\lambda}\mu$
0
1
2
3
$\cdots$

Fig. 2. Markov chain representing the size of a virtual queue. Here $\bar{\lambda} := (1 - \lambda)$ and $\bar{\mu} := (1 - \mu)$.

incorporates the current slot's reception also. Based on this feedback, packets are dropped from the physical queue *just before the end of the slot*, according to the queue update rule. Queue sizes are measured at the end of the slot.

F. Asymptotic Behavior

The load factor is denoted by $\rho := \lambda/\mu$. In what follows, we will study the asymptotic behavior of the expected queue size and decoding delay under various policies, as $\rho \to 1$ from below. For the asymptotics, we assume that either λ or μ is fixed, while the other varies causing ρ to increase to 1. We use the commonly-used asymptotic notations of $O(.)$, $\Theta(.)$ and $\Omega(.)$ to denote asymptotic bounds. Specifically, $f(x) = O(g(x))$ denotes that $g(x)$ bounds $f(x)$ asymptotically from above. $f(x) = \Omega(g(x))$ denotes that $g(x)$ bounds $f(x)$ asymptotically from below. $f(x) = \Theta(g(x))$ denotes that $g(x)$ bounds $f(x)$ asymptotically from above and below.

IV. Queue Size

In this section, we are concerned with minimizing the sender's physical queue size. We first present a baseline queue update algorithm – retain packets in the queue until the feedback confirms that they have been decoded by all the receivers. Then, we present a new queue update rule that is motivated by a novel coding algorithm. The new rule allows the physical queue size to track the virtual queue sizes.

A. Algorithm 1: Drop When Decoded (Baseline)

The baseline scheme combines a random coding strategy with a drop-when-decoded rule for queue update. The coding scheme is an online version of [8] with no preset generation size – a coded packet is formed by computing a random linear combination of all packets currently in the queue. With such a scheme, the innovation guarantee property will hold with high probability, provided the field size is large enough (We assume the field size is large enough to ignore the probability that the coded packet is not innovative. It can be incorporated into the model by assuming a slightly larger probability of erasure because a non-innovative packet is equivalent to an erasure.).

For any receiver, the packets at the sender are unknowns, and each received linear combination is an equation in these unknowns. Decoding becomes possible whenever the number of linearly independent equations catches up with the number of unknowns involved. The difference between the number of unknowns and the number of equations is essentially the backlog in degrees of freedom, *i.e.*, the virtual queue size. Thus, *a virtual queue becoming empty translates to successful decoding at the corresponding receiver*. Whenever a receiver is able to decode in this manner, it informs the sender. Based on this, the sender tracks which receivers have decoded each packet, and drops a packet if it has been decoded by all receivers. From a reliability perspective, this is fine because there is no need to involve decoded packets in the linear combination.

Remark 1: In general, it may be possible to solve for some of the unknowns even before the virtual queue becomes empty. For example, this could happen if a newly received linear combination cancels all but one unknown in a previously known linear combination. It could also happen if some packets were involved in a subset of equations that can be solved among themselves locally. Then, even if the overall system has more unknowns than equations, the packets involved in the local system can be decoded. However, these are secondary effects and we ignore them in this analysis. Equivalently, we assume that even if a packet is decoded before the virtual queue becomes empty, the sender waits for the next emptying of the virtual queue before dropping the packet. We believe this assumption will not change the asymptotic behavior of the queue size, since decoding before the virtual queue becoming empty is a rare event with random linear coding over a large field.

We will now compare the virtual and physical queue sizes in steady state. We introduce some notation:

$Q(t)$:= Sender's physical queue size at the end of slot t

$Q_j(t)$:= Size of jth virtual queue at the end of slot t

1) The Virtual Queue Size in Steady State: Figure 2 shows the Markov chain for $Q_j(t)$. If $\lambda < \mu$, then the chain $\{Q_j(t)\}$ is positive recurrent and has a steady state distribution given by [35]:

$$\pi_k := \lim_{t\to\infty} \mathbb{P}[Q_j(t) = k] = (1-\alpha)\alpha^k, \quad k \geq 0 \tag{1}$$

where $\alpha = \frac{\lambda(1-\mu)}{\mu(1-\lambda)}$. Thus, the expected size of any virtual queue in steady state is given by:

$$\lim_{t\to\infty} \mathbb{E}[Q_j(t)] = \sum_{j=0}^{\infty} j\pi_j = (1-\mu)\cdot\frac{\rho}{(1-\rho)} \tag{2}$$

2) The Physical Queue Size in Steady State: The following theorem characterizes the asymptotic behavior of the physical queue size under Algorithm 1, as the load on the system approaches capacity ($\rho \to 1$).

TABLE II
THE UNCODED VS. CODED CASE

	Uncoded Networks	**Coded Networks**
Knowledge represented by	Set of received packets	Vector space spanned by the coefficient vectors of the received linear combinations
Amount of knowledge	Number of packets received	Number of linearly independent (innovative) linear combinations of packets received (*i.e.*, dimension of the knowledge space)
Queue stores	All undelivered packets	Linear combination of packets which form a basis for the *coset space* of the common knowledge at all receivers
Update rule after each transmission	If a packet has been received by all receivers, drop it.	Recompute the common knowledge space V_Δ; Store a new set of linear combinations so that their span is independent of V_Δ

Theorem 1: The expected size of the physical queue in steady state for Algorithm 1 is $\Omega\left(\frac{1}{(1-\rho)^2}\right)$.

See Appendix A for the proof.

Comparing with Equation (2), this result makes it clear that the physical queue size under Algorithm 1 does not track the virtual queue size.

B. Algorithm 2 (a): Drop Common Knowledge

In this section, we first present a generic algorithm that operates at the level of knowledge spaces and their bases, in order to ensure that the physical queue size tracks the virtual queue size. Later, we shall describe a simple-to-implement variant of this generic algorithm.

1) An Intuitive Description: The aim of this algorithm is that the sender should store just enough data in its buffer so that it can always compute a linear combination which is simultaneously innovative to all receivers who have an information deficit, *i.e.*, so that it can ensure reliability and the innovation guarantee property.

After each slot, every receiver informs the sender whether an erasure occurred using perfect feedback. The sender then excludes from the queue, any knowledge that is common to all the receivers. Specifically, the queue's contents correspond to some basis of a vector space that is *independent of the intersection of the knowledge spaces of all the receivers.* We show in Lemma 2 that with this queuing rule, it is always possible to compute a linear combination of the current contents of the queue that will guarantee innovation, as long as the field size is more than n, the number of receivers.

The fact that the common knowledge is dropped suggests an incremental approach to the sender's operations. Although the knowledge spaces of the receivers grow with time, the sender only needs to operate with the projection of these spaces on dimensions currently in the queue, since the coding module does not care about the remaining part of the knowledge spaces that is common to all receivers. Thus, the algorithm can be implemented in an incremental manner. It will be shown that this incremental approach is equivalent to the cumulative approach.

Table II shows the main correspondence between the uncoded case and the coded case. We now present the queue update algorithm formally. Then we show that under this algorithm, the physical queue size at the sender tracks the virtual queue size.

Definition 8 (Local Coefficient Vector): For a linear combination of packets, the local coefficient vector is a vector of the coefficients applied to the current contents of the sender's queue in order to generate that linear combination. Its length is equal to the length of the sender's queue.

Definition 9 (Global Coefficient Vector): For a linear combination of packets, the global coefficient vector is a vector of the coefficients applied to global stream of packets in order to generate that linear combination. Its length is the total number of packets that have arrived in the stream so far.

All operations in the algorithm occur over a finite field of size $q > n$. The basis of a node's knowledge space is stored as the rows of a basis matrix. The representation and all operations are in terms of local coefficient vectors and not global ones.

2) Formal Description of the Algorithm: In the following, $span(M)$ denotes the row space of matrix M. I_m denotes the identity matrix of size m.

<u>*Algorithm 2 (a)*</u>

1. Initialize basis matrices $B, B_1, \ldots, B_n$ to the empty matrix. These contain the bases of the incremental knowledge spaces of the sender and receivers in that order.
2. Initialize $\mathbf{g}$ to $\mathbf{0}$. $\mathbf{g}$ will hold the coefficient vector of the transmitted packet in each slot.
   In every time slot, do:
3. *Incorporate new arrivals:*
   Let a be the number of new packets that arrive at the beginning of the slot. Place these packets at the end of the queue. Suppose B has b rows already. Set B to I_{a+b} (B will always be an identity matrix). To make the number of columns of all matrices consistent (*i.e.*, equal to $a+b$), append a all-zero columns to each B_j.
4. *Transmission:*
   If B is not empty, update $\mathbf{g}$ to be any vector that is in $span(B)$, but not in $\cup_{\{j:B_j \subsetneq B\}} span(B_j)$.
   Lemma 2 shows that such a $\mathbf{g}$ exists. Let $\mathbf{y_1}, \mathbf{y_2}, \ldots \mathbf{y_Q}$ represent the current contents of the queue, where the queue size $Q = (a+b)$. Compute the linear combination $\sum_{i=1}^{Q} g_i \mathbf{y_i}$ and transmit it on the packet erasure broadcast channel. If B is empty, set $\mathbf{g}$ to $\mathbf{0}$ and transmit nothing.

5. *Incorporate feedback:*
   Once the feedback arrives, for every receiver $j = 1$ to n, do:

   If $\mathbf{g} \neq \mathbf{0}$ and the transmission was successfully received by receiver j in this slot, append $\mathbf{g}$ as a new row to B_j.
6. *Separate out the knowledge that is common to all receivers:*
   Compute the following (the set notation used here considers the matrices as a set of row vectors):
   B_Δ:= Any basis of $\cap_{j=1}^n span(B_j)$.
   B':= Completion of B_Δ into a basis of $span(B)$.
   B'':= $B' \backslash B_\Delta$.
   B'_j := Completion of B_Δ into a basis of $span(B_j)$ in such a way that, B''_j defined as $B'_j \backslash B_\Delta$ satisfies: $B''_j \subseteq span(B'')$. Lemma 1 proves that this is possible.
7. *Update the queue contents:*
   Replace the contents of the queue with packets $\mathbf{y'_1}, \mathbf{y'_2}, \ldots \mathbf{y'_{Q'}}$ of the form $\sum_{i=1}^{Q} h_i \mathbf{y_i}$ for each $\mathbf{h} \in B''$. The new queue size Q' is thus equal to the number of rows in B''.
8. *Recompute local coefficient vectors with respect to the new queue contents:*
   Find a matrix X_j such that $B''_j = X_j B''$ (this is possible because $B''_j \subseteq span(B'')$). Call X_j the new B_j. Update the value of B to $I_{Q'}$.
9. Go back to step 3 for the next slot.

The above algorithm essentially removes at the end of each slot, the common knowledge (represented by the basis B_Δ) and retains only the remainder B''. The knowledge spaces of the receivers are also represented in an incremental manner in the form of B''_j, excluding the common knowledge. Since $B''_j \subseteq span(B'')$, the B''_j vectors can be completely described in terms of the vectors in B''. It is as if B_Δ has been completely removed, and the sender's goal is only to convey $span(B'')$ to the receivers. Hence, it is sufficient to store linear combinations corresponding to B'' in the queue. B'' and B''_j get mapped to the new B and B_j, and the process repeats in the next slot.

Lemma 1: In step 5 of the algorithm above, it is possible to complete B_Δ into a basis B'_j of each $span(B_j)$ such that $B''_j \subseteq span(B'')$.

See Appendix C for the proof.

Lemma 2 [20]: Let $\mathcal{V}$ be a vector space with dimension k over a field of size q, and let $\mathcal{V}_1, \mathcal{V}_2, \ldots \mathcal{V}_n$, be subspaces of $\mathcal{V}$, of dimensions $k_1, k_2, \ldots, k_n$ respectively. Suppose that $k > k_i$ for all $i = 1, 2, \ldots, n$. Then, there exists a vector that is in $\mathcal{V}$ but is not in any of the $\mathcal{V}_i$'s, if $q > n$.

Proof: See [20] for the proof. ■

This lemma is also closely related to the result in [30], which derives the smallest field size needed to ensure innovation guarantee.

3) Connecting the Physical and Virtual Queue Sizes: In this subsection, we will prove the following result that relates the size of the physical queue at the sender and the virtual queues, which themselves correspond to the backlog in degrees of freedom.

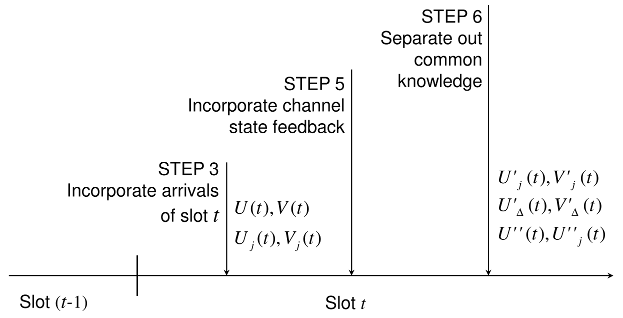
STEP 6
Separate out common knowledge
STEP 5
Incorporate channel state feedback
STEP 3
Incorporate arrivals of slot t
$U(t), V(t)$
$U_j(t), V_j(t)$
$U'_j(t), V'_j(t)$
$U'_\Delta(t), V'_\Delta(t)$
$U''(t), U''_j(t)$
Slot (t-1)
Slot t

Fig. 3. The main steps of the algorithm, along with the times at which the various $U(t)$'s are defined

Theorem 2: For Algorithm 2 (a), the physical queue size at the sender is upper bounded by the sum of the backlog differences between the sender and each receiver in terms of the number of degrees of freedom.

Let $a(t)$ denote the number of arrivals in slot t, and let $A(t)$ be the total number of arrivals up to and including slot t, *i.e.*, $A(t) = \sum_{t'=0}^{t} a(t')$. Let $B(t)$ (resp. $B_j(t)$) be the matrix B (resp. B_j) after incorporating the slot t arrivals, *i.e.*, at the end of step 3 in slot t. Let $H(t)$ be a matrix whose rows are the *global* coefficient vectors of the queue contents at the end of step 3 in slot t, *i.e.*, the coefficient vectors in terms of the original packet stream. Note that each row of $H(t)$ is in $\mathbb{F}_q^{A(t)}$.

Let $\mathbf{g}(t)$ denote the vector $\mathbf{g}$ calculated in step 4 in slot t, *i.e.*, the local coefficient vector of the packet transmitted in slot t. Also, let $B_\Delta(t)$ (resp. $B''(t)$, $B'_j(t)$ and $B''_j(t)$) denote the matrix B_Δ (resp. B'', B'_j and B''_j) at the end of step 6 in slot t.

Lemma 3: The rows of $H(t)$ are linearly independent for all t.

See Appendix D for the proof.

Define the following:
$U(t)$:= Row span of $H(t)$
$U_j(t)$:= Row span of $B_j(t)H(t)$
$U'_j(t)$:= Row span of $B'_j(t)H(t)$
$U'_\Delta(t) := \cap_{j=1}^n U'_j(t)$
$U''(t)$:= Row span of $B''(t)H(t)$
$U''_j(t)$:= Row span of $B''_j(t)H(t)$

All the vector spaces defined above are subspaces of $\mathbb{F}_q^{A(t)}$. Figure 3 shows the points at which these subspaces are defined in the slot.

The fact that $H(t)$ has full row rank (from Lemma 3) implies that the operations performed by the algorithm in the domain of the local coefficient vectors can be mapped to the corresponding operations in the domain of the global coefficient vectors:

1) The intersection subspace $U'_\Delta(t)$ is indeed the row span of $B_\Delta(t)H(t)$.
2) Let $R_j(t)$ be an indicator (0-1) random variable which takes the value 1 iff the transmission in slot t is successfully received without erasure by receiver j and in addition, receiver j does not have all the information that the sender has. Let $\tilde{\mathbf{g}}_\mathbf{j}(t) := R_j(t)\mathbf{g}(t)H(t)$. Then,

$$U'_j(t) = U_j(t) \oplus span(\tilde{\mathbf{g}}_\mathbf{j}(t)) \quad (3)$$

where $\oplus$ denotes direct sum of vector spaces. The way the algorithm chooses $\mathbf{g}(t)$ guarantees that if $R_j(t)$ is non-zero, then $\tilde{\mathbf{g}}_\mathbf{j}(t)$ will be outside the corresponding $U_j(t)$, *i.e.*, it will be innovative. This fact is emphasized by the direct sum in this equation.

3) Because of the way the algorithm performs the completion of the bases in the local domain in step 6, the following properties hold in the global domain:

$$U(t) = U'_\Delta(t) \oplus U''(t) \tag{4}$$
$$U'_j(t) = U'_\Delta(t) \oplus U''_j(t) \tag{5}$$

and,

$$U''_j(t) \subseteq U''(t), \quad \forall j = 1, 2, \ldots, n \tag{6}$$

From the above properties, we can infer that $U''_1(t)+U''_2(t)+\ldots U''_n(t) \subseteq U''(t)$. After incorporating the arrivals in slot $t+1$, this gives $U_1(t+1) + U_2(t+1) + \ldots U_n(t+1) \subseteq U(t+1)$. Since this is true for all t, we write it as:

$$U_1(t) + U_2(t) + \ldots U_n(t) \subseteq U(t) \tag{7}$$

Now, in order to relate the queue size to the backlog in number of degrees of freedom, we define the following vector spaces which represent the *cumulative* knowledge of the sender and receivers (See Figure 3 for the timing):

$V(t)$:= Sender's knowledge space after incorporating the arrivals (at the end of step 3) in slot t. This is simply equal to $\mathbb{F}_q^{A(t)}$

$V_j(t)$:= Receiver j's knowledge space at the end of step 3 in slot t

$V'_j(t)$:= Receiver j's knowledge space in slot t, after incorporating the channel state feedback into $V_j(t)$, *i.e.*, $V'_j(t) = V_j(t) \oplus span(\tilde{\mathbf{g}}_\mathbf{j}(t))$.

$V_\Delta(t) := \cap_{j=1}^n V_j(t)$

$V'_\Delta(t) := \cap_{j=1}^n V'_j(t)$

For completeness, we now prove the following facts about direct sums of vector spaces that we will use.

Lemma 4: Let V be a vector space and let $V_\Delta, U_1, U_2, \ldots U_n$ be subspaces of V such that, V_Δ is independent of the span of all the U_j's, i.e., $dim[V_\Delta \cap (U_1 + U_2 + \ldots + U_n)] = 0$. *Then,*

$$V_\Delta \oplus \left[\cap_{i=1}^n U_i\right] = \cap_{i=1}^n \left[V_\Delta \oplus U_i\right]$$

See Appendix E for the proof.

Lemma 5: Let A, B, and C be three vector spaces such that B is independent of C and A is independent of $B \oplus C$. Then the following hold:

1) *A is independent of B.*
2) *$A \oplus B$ is independent of C.*
3) *$A \oplus (B \oplus C) = (A \oplus B) \oplus C$.*

See Appendix F for the proof.

Theorem 3: For all $t \geq 0$,

$$V(t) = V_\Delta(t) \oplus U(t)$$
$$V_j(t) = V_\Delta(t) \oplus U_j(t) \quad \forall j = 1, 2, \ldots n$$
$$V'_\Delta(t) = V_\Delta(t) \oplus U'_\Delta(t)$$

Proof: The proof is by induction on t.

Basis step:

At $t = 0$, $V(0)$, $U(0)$ as well as all the $V_j(0)$'s and $U_j(0)$'s are initialized to $\{\mathbf{0}\}$. Consequently, $V_\Delta(0)$ is also $\{\mathbf{0}\}$. It is easily seen that these initial values satisfy the equations in the theorem statement.

Induction Hypothesis:

We assume the equations hold at t, *i.e.*,

$$V(t) = V_\Delta(t) \oplus U(t) \tag{8}$$
$$V_j(t) = V_\Delta(t) \oplus U_j(t), \quad \forall j = 1, 2, \ldots n \tag{9}$$
$$V'_\Delta(t) = V_\Delta(t) \oplus U'_\Delta(t) \tag{10}$$

Induction Step: We now prove that they hold in slot $(t+1)$. We have:

$$\begin{aligned} V(t) &= V_\Delta(t) \oplus U(t) \quad \text{(from (8))} \\ &= V_\Delta(t) \oplus [U'_\Delta(t) \oplus U''(t)] \quad \text{(from (4))} \\ &= [V_\Delta(t) \oplus U'_\Delta(t)] \oplus U''(t) \quad \text{(Lemma 5)} \\ &= V'_\Delta(t) \oplus U''(t) \quad \text{(from (10))} \end{aligned}$$

Thus, we have proved:

$$V(t) = V'_\Delta(t) \oplus U''(t) \tag{11}$$

Now, we incorporate the arrivals in slot $(t+1)$. This converts $V'_\Delta(t)$ to $V_\Delta(t+1)$, $U''(t)$ to $U(t+1)$, and $V(t)$ to $V(t+1)$, due to the following operations:

$$\text{Basis of } V_\Delta(t+1) = \begin{bmatrix}\text{Basis of } V'_\Delta(t) & 0\end{bmatrix}$$
$$\text{Basis of } U(t+1) = \begin{bmatrix}\text{Basis of } U''(t) & 0 \\ 0 & I_{a(t+1)}\end{bmatrix}$$
$$\text{Basis of } V(t+1) = \begin{bmatrix}\text{Basis of } V(t) & 0 \\ 0 & I_{a(t+1)}\end{bmatrix}$$

Incorporating these modifications into (11), we get:

$$V(t+1) = V_\Delta(t+1) \oplus U(t+1)$$

Now, consider each receiver $j = 1, 2, \ldots n$.

$$\begin{aligned} V'_j(t) &= V_j(t) \oplus span(\tilde{\mathbf{g}}_\mathbf{j}(t)) \\ &= [V_\Delta(t) \oplus U_j(t)] \oplus span(\tilde{\mathbf{g}}_\mathbf{j}(t)) \quad \text{(from (9))} \\ &= V_\Delta(t) \oplus [U_j(t) \oplus span(\tilde{\mathbf{g}}_\mathbf{j}(t))] \quad \text{(Lemma 5)} \\ &= V_\Delta(t) \oplus U'_j(t) \quad \text{(from (3))} \\ &= V_\Delta(t) \oplus [U'_\Delta(t) \oplus U''_j(t)] \quad \text{(from (5))} \\ &= [V_\Delta(t) \oplus U'_\Delta(t)] \oplus U''_j(t) \quad \text{(Lemma 5)} \\ &= V'_\Delta(t) \oplus U''_j(t) \quad \text{(from (10))} \end{aligned}$$

Incorporating the new arrivals into the subspaces involves adding $a(t+1)$ all-zero columns to the bases of $V'_j(t)$, $V'_\Delta(t)$, and $U''_j(t)$, thereby converting them into bases of $V_j(t+1)$, $V_\Delta(t+1)$, and $U_j(t+1)$ respectively. These changes do not affect the above relation, and we get:

$$V_j(t+1) = V_\Delta(t+1) \oplus U_j(t+1), \quad \forall j = 1, 2, \ldots n$$

And finally,

$$\begin{aligned}
&V'_{\Delta}(t+1)\\
&= \cap_{j=1}^{n} V'_j(t+1)\\
&= \cap_{j=1}^{n}[V_j(t+1) \oplus span(\tilde{\mathbf{g}}_{\mathbf{j}}(t+1))]\\
&= \cap_{j=1}^{n}[V_{\Delta}(t+1) \oplus U_j(t+1) \oplus span(\tilde{\mathbf{g}}_{\mathbf{j}}(t+1))]\\
&\overset{(a)}{=} V_{\Delta}(t+1) \oplus \cap_{j=1}^{n}[U_j(t+1) \oplus span(\tilde{\mathbf{g}}_{\mathbf{j}}(t+1))]\\
&= V_{\Delta}(t+1) \oplus U'_{\Delta}(t+1)
\end{aligned}$$

Step (a) is justified as follows. Using equation (7) and the fact that $\tilde{\mathbf{g}}_{\mathbf{j}}(t+1)$ was chosen to be inside $U(t+1)$, we can show that the span of all the $[U_j(t+1) \oplus span(\tilde{\mathbf{g}}_{\mathbf{j}}(t+1))]$'s is inside $U(t+1)$. Now, from the induction step above, $V_{\Delta}(t+1)$ is independent of $U(t+1)$. Therefore, $V_{\Delta}(t+1)$ is independent of the span of all the $[U_j(t+1) \oplus span(\tilde{\mathbf{g}}_{\mathbf{j}}(t+1))]$'s. We can therefore apply Lemma 4. ∎

Theorem 4: Let $Q(t)$ denote the size of the queue after the arrivals in slot t have been appended to the queue.

$$Q(t) = dim\ V(t) - dim\ V_{\Delta}(t)$$

Proof:

$$\begin{aligned}
Q(t) &= dim\ U(t) = dim\ U''(t-1) + a(t)\\
&= dim\ U(t-1) - dim\ U'_{\Delta}(t-1) + a(t)\\
&\qquad \text{(using (4))}\\
&= dim\ V(t-1) - dim\ V_{\Delta}(t-1) - dim\ U'_{\Delta}(t) + a(t)\\
&\qquad \text{(from Theorem 3)}\\
&= dim\ V(t-1) - dim\ V'_{\Delta}(t) + a(t)\\
&\qquad \text{(from Theorem 3)}\\
&= dim\ V(t) - dim\ V_{\Delta}(t)
\end{aligned}$$

∎

Lemma 6: Let $V_1, V_2, \ldots, V_k$ be subspaces of a vector space V. Then, for $k \geq 1$,

$$dim(V_1 \cap V_2 \cap \ldots \cap V_k) \geq \sum_{i=1}^{k} dim(V_i) - (k-1)dim(V)$$

See Appendix G for the proof. The above result can be rewritten as:

$$dim(V) - dim(V_1 \cap V_2 \cap \ldots V_k) \leq \sum_{i=1}^{k}[dim(V) - dim(V_i)] \tag{12}$$

Using this result, we can now prove Theorem 2.

Proof of Theorem 2: If we apply Lemma 6 to the vector spaces $V_j(t), j = 1, 2, \ldots, n$ and $V(t)$, then the left hand side of inequality (12) becomes the sender queue size (using Theorem 4), while the right hand side becomes the sum of the differences in backlog between the sender and the receivers, in terms of the number of degrees of freedom. Thus, we have proved Theorem 2. ∎

C. Algorithm 2 (b): Drop when seen

The drop-when-seen algorithm can be viewed as a specialized variant of Algorithm 2 (a). It uses the notion of seen packets (defined in Section II) to represent the bases of the knowledge spaces. This leads to a simple and easy-to-implement version of the algorithm which, besides ensuring that physical queue size tracks virtual queue size, also provides some practical benefits. For instance, the sender need not store linear combinations of packets in the queue like in Algorithm 2 (a). Instead only original packets need to be stored, and the queue can be operated in a simple first-in-first-out manner. We now present some mathematical preliminaries before describing the algorithm.

1) Some Preliminaries: Algorithm 2 (b) uses the notion of reduced row echelon form (RREF) of a matrix to represent the knowledge of a receiver. The definition and properties of the RREF can be found in [19]. We now present the connection between the RREF and the notion of seeing packets.

The RREF leads to a standard way to represent a vector space. Given a vector space, arrange the basis vectors in any basis of the space as the rows of a matrix, and perform Gauss-Jordan elimination. This process produces a unique matrix in RREF such that its row space is the given vector space. We call this the *RREF basis matrix* of the space. We will use this representation for the knowledge space of the receivers.

Let V be the knowledge space of some receiver. Suppose m packets have arrived at the sender so far. Then the receiver's knowledge consists of linear combinations of these m packets, *i.e.*, V is a subspace of $\mathbb{F}_q^m$. Using the procedure outlined above, we can compute the $dim(V) \times m$ RREF basis matrix of V over $\mathbb{F}_q$.

In the RREF basis, the first nonzero entry of any row is called a *pivot*. Any column with a pivot is called a *pivot column*. By definition, each pivot occurs in a different column. Hence, the number of pivot columns equals the number of nonzero rows, which is $dim[V]$. Let $\mathbf{p_k}$ denote the packet with index k. The columns are ordered so that column k maps to packet $\mathbf{p_k}$. The following theorem connects the notion of seeing packets to the RREF basis.

Theorem 5: A node has seen a packet with index k if and only if the kth column of the RREF basis B of the knowledge space V of the node is a pivot column.

Proof: The 'if' part is clear. If column k of B is a pivot column, then the corresponding pivot row corresponds to a linear combination known to the node, of the form $\mathbf{p_k} + \mathbf{q}$, where $\mathbf{q}$ involves only packets with index more than k. Thus, the node has seen $\mathbf{p_k}$.

For the 'only if' part, suppose column k of B does not contain a pivot. Then, in any linear combination of the rows, rows with pivot after column k cannot contribute anything to column k. Rows with pivot before column k will result in a non-zero term in some column to the left of k. Since every vector in V is a linear combination of the rows of B, the first non-zero term of any vector in V cannot be in column k. Thus, $\mathbf{p_k}$ could not have been seen. ∎

Since the number of pivot columns is equal to the dimension of the vector space, we obtain the following corollary.

Corollary 1: The number of packets seen by a receiver is equal to the dimension of its knowledge space.

The next corollary introduces a useful concept.

Corollary 2: If receiver j has seen packet $\mathbf{p_k}$, then it knows exactly one linear combination of the form $\mathbf{p_k} + \mathbf{q}$ such that $\mathbf{q}$ involves only **unseen** *packets with index more than k.*

Proof: We use the same notation as above. The receiver has seen $\mathbf{p_k}$. Hence, column k in B is a pivot column. By definition of RREF, in the row containing the pivot in column k, the pivot value is 1 and subsequent nonzero terms occur only in non-pivot columns. Thus, the corresponding linear combination has the given form $\mathbf{p_k}+\mathbf{q}$, where $\mathbf{q}$ involves only *unseen* packets with index more than k.

We now prove uniqueness by contradiction. Suppose the receiver knows another such linear combination $\mathbf{p_k} + \mathbf{q}'$ where $\mathbf{q}'$ also involves only unseen packets. Then, the receiver must also know $(\mathbf{q} - \mathbf{q}')$. But this means the receiver has seen some packet involved in either $\mathbf{q}$ or $\mathbf{q}'$ – a contradiction. ■

Definition 10 (Witness): We denote the unique linear combination guaranteed by Corollary 2 as $\mathbf{W_j}(\mathbf{p_k})$, the *witness for receiver j seeing $\mathbf{p_k}$*.

2) Description of Algorithm 2 (b): The central idea of the algorithm is to keep track of seen packets instead of decoded packets. The two main parts of the algorithm are the coding and queue update modules.

In Section IV-C5, we present the formal description of our coding module. The coding module computes a linear combination $\mathbf{g}$ that will cause any receiver that receives it, to see its next unseen packet. First, for each receiver, the sender computes its knowledge space using the feedback and picks out its next unseen packet. Only these packets will be involved in $\mathbf{g}$, and hence we call them the *transmit set*. Now, we need to select coefficients for each packet in this set. Clearly, the receiver(s) waiting to see the oldest packet in the transmit set (say $\mathbf{p_1}$) will be able to see it as long as its coefficient is not zero. Consider a receiver that is waiting to see the second oldest packet in the transmit set (say $\mathbf{p_2}$). Since the receiver has already seen $\mathbf{p_1}$, it can subtract the witness for $\mathbf{p_1}$, thereby canceling it from $\mathbf{g}$. The coefficient of $\mathbf{p_2}$ must be picked such that after subtracting the witness for $\mathbf{p_1}$, the remaining coefficient of $\mathbf{p_2}$ in $\mathbf{g}$ is non-zero. The same idea extends to the other coefficients. The receiver can cancel packets involved in $\mathbf{g}$ that it has already seen by subtracting suitable multiples of the corresponding witnesses. Therefore, the coefficients for $\mathbf{g}$ should be picked such that for each receiver, after canceling the seen packets, the remaining coefficient of the next unseen packet is non-zero. Then, the receiver will be able to see its next unseen packet. Theorem 8 proves that this is possible if the field size is at least n, the number of receivers. With two receivers, the coding module is a simple XOR based scheme (see Table I). Our coding scheme meets the innovation guarantee requirement because Theorem 5 implies that a linear combination that would cause a new packet to be seen brings in a previously unknown degree of freedom.

The fact that the coding module uses only the next unseen packet of all receivers readily implies the following queue update rule. **Drop a packet if all receivers have seen it.** This simple rule ensures that the physical queue size tracks the virtual queue size.

Remark 2: In independent work, [30] proposes a coding algorithm which uses the idea of selecting those packets for coding, whose indices are one more than each receiver's rank. This corresponds to choosing the next unseen packets in the special case where packets are seen in order. Moreover, this algorithm picks coding coefficients in a deterministic manner, just like our coding module. Therefore, our module is closely related to the algorithm of [30].

However, our algorithm is based on the framework of seen packets. This allows several benefits. First, it immediately leads to the drop-when-seen queue management algorithm, as described above. In contrast, [30] does not consider queuing aspects of the problem. Second, in this form, our algorithm readily generalizes to the case where the coding coefficients are picked randomly. The issue with random coding is that packets may be seen out of order. Our algorithm will guarantee innovation even in this case (provided the field is large), by selecting a random linear combination of the next unseen packets of the receivers. However, the algorithm of [30] may not work well here, as it may pick packets that have already been seen, which could cause non-innovative transmissions.

The compatibility of our algorithm with random coding makes it particularly useful from an implementation perspective. With random coding, each receiver only needs to inform the sender the set of packets it has seen. There is no need to convey the exact knowledge space. This can be done simply by generating a TCP-like cumulative ACK upon seeing a packet. Thus, the ACK format is the same as in traditional ARQ-based schemes. Only its interpretation is different.

We next present the formal description and analysis of the queue update algorithm.

3) The Queuing Module: The algorithm works with the RREF bases of the receivers' knowledge spaces. The coefficient vectors are with respect to the current queue contents and not the original packet stream.

Algorithm 2 (b)

```
1. Initialize matrices $B_1, B_2, \ldots, B_n$ to the empty matrix.
   These matrices will hold the bases of the incremental
   knowledge spaces of the receivers.
2. Incorporate new arrivals: Suppose there are $a$ new
   arrivals. Add the new packets to the end of the queue.
   Append $a$ all-zero columns on the right to each $B_j$ for
   the new packets.
3. Transmission: If the queue is empty, do nothing; else
   compute $\mathbf{g}$ using the coding module and transmit it.
4. Incorporate channel state feedback:
   For every receiver $j = 1$ to $n$, do:
   If receiver $j$ received the transmission, include the
   coefficient vector of $\mathbf{g}$ in terms of the current queue
   contents, as a new row in $B_j$. Perform Gauss-Jordan
   elimination.
5. Separate out packets that all receivers have seen:
   Update the following sets and bases:
     $S'_j$ := Set of packets corresponding to the pivot
   columns of $B_j$
     $S'_\Delta := \cap_{j=1}^{n} S'_j$
```

New B_j := Sub-matrix of current B_j obtained by excluding columns in S'_Δ and corresponding pivot rows.

6. *Update the queue:* Drop the packets in S'_Δ.
7. Go back to step 2 for the next slot.

4) Connecting the Physical and Virtual Queue Sizes:

Theorem 6: For Algorithm 2 (b), the physical queue size at the sender is upper-bounded by the sum of the virtual queue sizes, i.e., *the sum of the degrees-of-freedom backlog between the sender and the receivers. Hence, the expected size of the physical queue in steady state for Algorithm 2 (b) is* $O\left(\frac{1}{1-\rho}\right)$.

In the rest of this section, we will prove the above result. Now, in order to relate the queue size to the backlog in number of degrees of freedom, we will need the following notation:

$S(t)$:= Set of packets arrived at sender till the end of slot t

$V(t)$:= Sender's knowledge space after incorporating the arrivals in slot t. This is simply equal to $\mathbb{F}_q^{|S(t)|}$

$V_j(t)$:= Receiver j's knowledge space at the end of slot t. It is a subspace of $V(t)$.

$S_j(t)$:= Set of packets receiver j has seen till end of slot t

We will now formally argue that Algorithm 2 (b) indeed implements the drop-when-seen rule in spite of the incremental implementation. In any slot, the columns of B_j are updated as follows. When new packets are appended to the queue, new columns are added to B_j on the right. When packets are dropped from the queue, corresponding columns are dropped from B_j. There is no rearrangement of columns at any point. This implies that a one-to-one correspondence is always maintained between the columns of B_j and the packets currently in the queue. Let $U_j(t)$ be the row space of B_j at time t. Thus, if $(u_1, u_2, \ldots, u_{Q(t)})$ is any vector in $U_j(t)$, it corresponds to a linear combination of the form $\sum_{i=1}^{Q(t)} u_i\mathbf{p_i}$, where $\mathbf{p_i}$ is the ith packet in the queue at time t. The following theorem connects the incremental knowledge space $U_j(t)$ to the cumulative knowledge space $V_j(t)$.

Theorem 7: In Algorithm 2 (b), for each receiver j, at the end of slot t, for any $\mathbf{u} \in U_j(t)$, the linear combination $\sum_{i=1}^{Q(t)} u_i\mathbf{p_i}$ is known to receiver j, where $\mathbf{p_i}$ denotes the ith packet in the queue at time t.

Proof: We will use induction on t. For $t = 0$, the system is completely empty and the statement is vacuously true. Let us now assume that the statement is true at time $(t-1)$. Consider the operations in slot t. A new row is added to B_j only if the corresponding linear combination has been successfully received by receiver j. Hence, the statement is still true. Row operations involved in Gauss-Jordan elimination do not alter the row space. Finally, when some of the pivot columns are dropped along with the corresponding pivot rows in step 5, this does not affect the linear combinations to which the remaining rows correspond because the pivot columns have a 0 in all rows except the pivot row. Hence, the three operations that are performed between slot $(t-1)$ and slot t do not affect the property that the vectors in the row space of B_j correspond to linear combinations that are known at receiver j. This proves the theorem. ■

If a packet corresponds to a pivot column in B_j, the corresponding pivot row is a linear combination of the packet in question with packets that arrived after it. From the above theorem, receiver j knows this linear combination which means it has seen the packet. This leads to the following corollary.

Corollary 3: If a packet corresponds to a pivot column in B_j, then it has been seen by receiver j.

Thus, in step 5, $S'_\Delta(t)$ consists of those packets in the queue that all receivers have seen by the end of slot t. In other words, the algorithm retains only those packets that have not yet been seen by all receivers. Even though the algorithm works with an incremental version of the knowledge spaces, namely $U_j(t)$, it maintains the queue in the same way as if it was working with the cumulative version $V_j(t)$. Thus, the incremental approach is equivalent to the cumulative approach.

We will require the following lemma to prove the main theorem.

Lemma 7: Let $A_1, A_2, \ldots, A_k$ be subsets of a set A. Then, for $k \geq 1$,

$$|A| - |\cap_{i=1}^k A_i| \leq \sum_{i=1}^k (|A| - |A_i|) \tag{13}$$

Proof:

$$\begin{aligned}
&|A| - |\cap_{i=1}^k A_i| \\
&= |A \cap (\cap_{i=1}^k A_i)^c| \quad \text{(since the } A_i\text{'s are subsets of } A) \\
&= |A \cap (\cup_{i=1}^k A_i^c)| \quad \text{(by De Morgan's law)} \\
&= |\cup_{i=1}^k (A \cap A_i^c)| \quad \text{(distributivity)} \\
&\leq \sum_{i=1}^k |A \cap A_i^c| \quad \text{(union bound)} \\
&= \sum_{i=1}^k (|A| - |A_i|)
\end{aligned}$$

■

Now, we are ready to prove Theorem 6.

Proof of Theorem 6: Since the only packets in the queue at any point are those that not all receivers have seen, we obtain the following expression for the physical queue size at the sender at the end of slot t:

$$Q(t) = |S(t)| - |\cap_{j=1}^n S_j(t)|$$

If we apply Lemma 7 to the sets $S(t)$ and $S_j(t)$, $j = 1, 2, \ldots, n$ then the left hand side of inequality (13) becomes the sender queue size $Q(t)$ given above. Now, $|S_j(t)| = dim[V_j(t)]$, using Corollary 1. Hence the right hand side of inequality (13) can be rewritten as $\sum_{j=1}^n \left[dim[V(t)] - dim[V_j(t)]\right]$, which is the sum of the virtual queue sizes.

Finally, we can find the asymptotic behavior of the physical queue size in steady state under Algorithm 2 (b). Since the expected virtual queue sizes themselves are all $O\left(\frac{1}{1-\rho}\right)$ from Equation (2), we obtain the stated result. ■

5) The Coding Module: We now present a coding module that is compatible with the drop-when-seen queuing algorithm in the sense that it always forms a linear combination using packets that are currently in the queue maintained by the queuing module. In addition, we show that the coding module satisfies the innovation guarantee property.

Let $\{u_1, u_2, \ldots, u_m\}$ be the set of indices of the next unseen packets of the receivers, sorted in ascending order (In general,

$m \leq n$, since the next unseen packet may be the same for some receivers). Exclude receivers whose next unseen packets have not yet arrived at the sender. Let $R(u_i)$ be the set of receivers whose next unseen packet is $\mathbf{p_{u_i}}$. We now present the coding module to select the linear combination for transmission.

1) *Loop over next unseen packets*
For $j = 1$ to m, do:
All receivers in $R(u_j)$ have seen packets $\mathbf{p_{u_i}}$ for $i < j$. Now, $\forall r \in R(u_j)$, find $\mathbf{y_r} := \sum_{i=1}^{j-1} \alpha_i \mathbf{W_r}(\mathbf{p_{u_i}})$, where $\mathbf{W_r}(\mathbf{p_{u_i}})$ is the witness for receiver r seeing $\mathbf{p_{u_i}}$. Pick $\alpha_j \in \mathbb{F}_q$ such that α_j is different from the coefficient of $\mathbf{p_{u_j}}$ in $\mathbf{y_r}$ for each $r \in R(u_j)$.

2) *Compute the transmit packet:* $\mathbf{g} := \sum_{i=1}^{m} \alpha_i \mathbf{p_{u_i}}$

It is easily seen that this coding module is compatible with the drop-when-seen algorithm. Indeed, it does not use any packet that has been seen by all receivers in the linear combination. It only uses packets that at least one receiver has not yet seen. The queue update module retains precisely such packets in the queue. The next theorem presents a useful property of the coding module.

Theorem 8: If the field size is at least n, then the coding module picks a linear combination that will cause any receiver to see its next unseen packet upon successful reception.

Proof: First we show that a suitable choice always exists for α_j that satisfies the requirement in step 1. For $r \in R(u_1)$, $\mathbf{y_r} = \mathbf{0}$. Hence, as long as $\alpha_1 \neq 0$, the condition is satisfied. So, pick $\alpha_1 = 1$. Since at least one receiver is in $R(u_1)$, we have that for $j > 1$, $|R(u_j)| \leq (n-1)$. Even if each $\mathbf{y_r}$ for $r \in R(u_j)$ has a different coefficient for $\mathbf{p_{u_j}}$, that covers only $(n-1)$ different field elements. If $q \geq n$, then there is a choice left in $\mathbb{F}_q$ for α_j.

Now, we have to show that the condition given in step 1 implies that the receivers will be able to see their next unseen packet. Indeed, for all j from 1 to m, and for all $r \in R(u_j)$, receiver r knows $\mathbf{y_r}$, since it is a linear combination of witnesses of r. Hence, if r successfully receives $\mathbf{g}$, it can compute $(\mathbf{g} - \mathbf{y_r})$. Now, $\mathbf{g}$ and $\mathbf{y_r}$ have the same coefficient for all packets with index less than u_j, and a different coefficient for $\mathbf{p_{u_j}}$. Hence, $(\mathbf{g} - \mathbf{y_r})$ will involve $\mathbf{p_{u_j}}$ and only packets with index beyond u_j. This means r can see $\mathbf{p_{u_j}}$ and this completes the proof. ■

Theorem 5 implies that seeing an unseen packet corresponds to receiving an unknown degree of freedom. Thus, Theorem 8 essentially says that the innovation guarantee property is satisfied and hence the scheme is throughput optimal.

This theorem is closely related to the result derived in [30] that computes the minimum field size needed to guarantee innovation. The difference is that our result uses the framework of seen packets to make a more general statement by specifying not only that innovation is guaranteed, but also that packets will be seen in order with this deterministic coding scheme. This means packets will be dropped in order at the sender.

V. Overhead

In this section, we comment on the overhead required for Algorithms 1 and 2 (b). There are several types of overhead.

A. Amount of Feedback

Our scheme assumes that every receiver feeds back one bit after every slot, indicating whether an erasure occurred or not. In comparison, the drop-when-decoded scheme requires feedback only when packets get decoded. However, in that case, the feedback may be more than one bit – the receiver will have to specify the list of all packets that were decoded, since packets may get decoded in groups. In a practical implementation of the drop-when-seen algorithm, TCP-like cumulative acknowledgments can be used to inform the sender which packets have been seen.

B. Identifying the Linear Combination

Together with a linear combination of packets, a sender must specify which packets have been involved in the combination, and what coefficients were used for these packets.

1) Set of Packets Involved: The baseline algorithm uses all packets in the queue for the linear combination. The queue is updated in a first-in-first-out (FIFO) manner. This is a consequence of the fact that the receiver signals successful decoding only when the virtual queue becomes empty.[5] The FIFO rule implies that specifying the contents of the queue reduces to specifying the sequence number of the head-of-line packet and the last packet in the queue in every transmission.

The drop-when-seen algorithm does not use all packets from the queue, but at most n packets (the next unseen packet of each receiver). The sender could specify the set of packets involved by listing their sequence numbers. Here also the sender's queue follows a FIFO rule, since the coding module guarantees that packets will be seen in order.

In both cases, the sequence number of the original stream cannot be used as it is, since it grows with time. However, since the queues are updated in a FIFO manner, we can express the sequence number relative to an origin that also advances with time. If the sender knows that the receiver's estimate of the sender's queue starts at a particular point, then both the sender and receiver can reset their origin to that point and count from there.

For the baseline case, if the receiver has a decoding event in a particular slot, it must have had a successful reception in that slot. Hence, upon receiving feedback that indicates the successful decoding, the sender can be certain that the receiver must have received the latest update about the queue contents and is therefore in sync with the sender. At this point, both the sender and receiver can reset their origin to the current HOL packet. Since the decoding epochs of different receivers may not be synchronized, the sender will have to maintain a different origin for each receiver and send a different sequence number to each receiver, relative to that receiver's origin. This can be done simply by concatenating the sequence number for each receiver in the header. The range of values that the sequence number can take will be proportional to the busy period of the virtual queue, since this determines how often the origin is reset. Thus, the overhead in bits for each receiver

[5] As mentioned earlier in Remark 1, we assume that the sender checks whether any packets have been newly decoded only when the virtual queue becomes empty.

will be proportional to the logarithm of the expected busy period, *i.e.*, $O\left(\log_2 \frac{1}{1-\rho}\right)$.

For the drop-when-seen scheme, the origin can be reset whenever the receiver sends feedback indicating successful reception. Thus, the origin advances a lot more frequently than in the baseline scheme.

2) Coefficients Used: The baseline algorithm uses a random linear coding scheme. Here, potentially all packets in the queue get combined in a linear combination. So, in the worst case, the sender would have to send one coefficient for every packet in the queue. If the queue has m packets, this would require $m \log_2 q$ bits, where q is the field size. In expectation, this would be $O\left(\frac{\log_2 q}{(1-\rho)^2}\right)$ bits. If the receiver knows the pseudorandom number generator used by the sender, then it would be sufficient for the sender to send the current state of the generator and the size of the queue. Using this, the receiver can generate the coefficients used by the sender in the coding process. The new drop-when-seen algorithm uses a coding module which combines the next unseen packet of each receiver. Thus, the overhead for the coefficients is at most $n \log_2 q$ bits, where n is the number of receivers. It does not depend on the load factor ρ at all.

C. Overhead at Sender

While Algorithm 2 (b) saves in buffer space, it requires the sender to store the basis matrix of each receiver, and update them in every slot based on feedback. However, storing a row of the basis matrix requires much less memory than storing a packet, especially for long packets. Thus, there is an overall saving in memory. The update of the basis matrix simply involves one step of the Gauss-Jordan elimination algorithm.

D. Overhead at Receiver

The receiver will have to store the coded packets till they are decoded. It will also have to decode the packets. For this, the receiver can perform a Gauss-Jordan elimination after every successful reception. Thus, the computation for the matrix inversion associated with decoding can be spread over time.

VI. Decoding Delay

With the coding module of Section IV-C.5, although a receiver can see the next unseen packet in every successful reception, this does not mean the packet will be decoded immediately. In general, the receiver will have to collect enough equations in the unknown packets before being able to decode them, resulting in a delay. We consider two notions of delay in this paper:

Definition 11 (Decoding Delay): The *decoding delay* of a packet with respect to a receiver is the time that elapses between the arrival of the packet at the sender and the decoding of the packet by the receiver under consideration.

As discussed in Section I, some applications can make use of a packet only if all prior packets have been decoded. In other words, the application will accept packets only up to the front of contiguous knowledge. This motivates the following stronger notion of delay.

Definition 12 (Delivery Delay): The *delivery delay* of a packet with respect to a receiver is the time that elapses between the arrival of the packet at the sender and the delivery of the packet by the receiver to the application, with the constraint that packets may be delivered only in order.

It follows from these definitions that *the decoding delay is always less than or equal to the delivery delay*. Upon decoding the packets, the receiver will place them in a reordering buffer until they are delivered to the application.

In this section, we study the expectation of these delays for an arbitrary packet. It can be shown using ergodic theory that the long term average of the delay experienced by the packets in steady state converges to this expectation with high probability. We focus on the asymptotic growth of the expected delay as $\rho \to 1$.

The section is organized as follows. We first study the delivery delay behavior of Algorithms 1 and 2(b), and provide an upper bound on the asymptotic expected delivery delay for any policy that satisfies the innovation guarantee property. We then present a generic lower bound on the expected decoding delay. Finally, we present a new coding module which not only guarantees innovation, but also aims to minimize the delivery delay. We conjecture that this algorithm achieves a delivery delay whose asymptotic growth matches that of the lower bound. This behavior is verified through simulations.

A. Upper Bound on Delivery Delay

Theorem 9: The expected delivery delay of a packet for any coding module that satisfies the innovation guarantee property is $O\left(\frac{1}{(1-\rho)^2}\right)$.

The arguments leading to this bound are presented next. For any policy that satisfies the innovation guarantee property, the virtual queue size evolves according to the Markov chain in Figure 2. The analysis of Algorithm 1 in Section IV-A therefore applies to any coding algorithm that guarantees innovation.

As explained in that section, the event of a virtual queue becoming empty translates to successful decoding at the corresponding receiver, since the number of equations now matches the number of unknowns involved. Thus, an arbitrary packet that arrives at the sender will get decoded by receiver j at or before the next emptying of the jth virtual queue. In fact, it will get delivered to the application at or before the next emptying of the virtual queue. This is because, when the virtual queue is empty, every packet that arrived at the sender gets decoded. Thus, the front of contiguous knowledge advances to the last packet that the sender knows.

The above discussion implies that Equation (14) gives an upper bound on the expected delivery delay of an arbitrary packet. We thus obtain the result stated above.

We next study the decoding delay of Algorithm 2 (b). We define *the decoding event* to be the event that all seen packets get decoded. Since packets are always seen in order, the decoding event guarantees that the front of contiguous knowledge will advance to the front of seen packets.

We use the term *leader* to refer to the receiver which has seen the maximum number of packets at the given point in time. Note that there can be more than one leader at the same time. The following theorem characterizes sufficient conditions for the decoding event to occur.

Theorem 10: The decoding event occurs in a slot at a particular receiver if in that slot:

(a) *The receiver has a successful reception which results in an empty virtual queue at the sender; OR*
(b) *The receiver has a successful reception and the receiver was a leader at the beginning of the slot.*

Proof: Condition (a) implies that the receiver has seen all packets that have arrived at the sender up to that slot. Each packet at the sender is an unknown and each seen packet corresponds to a linearly independent equation. Thus, the receiver has received as many equations as the number of unknowns, and can decode all packets it has seen.

Suppose condition (b) holds. Let $\mathbf{p_k}$ be the next unseen packet of the receiver in question. The sender's transmitted linear combination will involve only the next unseen packets of all the receivers. Since the receiver was a leader at the beginning of the slot, the sender's transmission will not involve any packet beyond $\mathbf{p_k}$, since the next unseen packet of all other receivers is either $\mathbf{p_k}$ or some earlier packet. After subtracting the suitably scaled witnesses of already seen packets from such a linear combination, the leading receiver will end up with a linear combination that involves only $\mathbf{p_k}$. Thus the leader not only sees $\mathbf{p_k}$, but also decodes it. In fact, none of the sender's transmissions so far would have involved any packet beyond $\mathbf{p_k}$. Hence, once $\mathbf{p_k}$ has been decoded, $\mathbf{p_{k-1}}$ can also be decoded. This procedure can be extended to all unseen packets, and by induction, we can show that all unseen packets will be decoded. ■

The upper bound proved in Theorem 9 is based on the emptying of the virtual queues. This corresponds only to case (a) in Theorem 10. The existence of case (b) shows that in general, the decoding delay will be strictly smaller than the upper bound. A natural question is whether this difference is large enough to cause a different asymptotic behavior, *i.e.*, does Algorithm 2 (b) achieve a delay that asymptotically has a smaller exponent of growth than the upper bound as $\rho \to 1$? We conjecture that this is not the case, *i.e.*, that the decoding delay for Algorithm 2 (b) is also $\Omega\left(\frac{1}{(1-\rho)^2}\right)$, although the constant of proportionality will be smaller. For the two receiver case, based on our simulations, this fact seems to be true. Figure 4 shows the growth of the decoding delay averaged over a large number of packets, as a function of $\frac{1}{(1-\rho)}$. The resulting curve seems to be close to the curve $\frac{0.37}{(1-\rho)^2}$, implying a quadratic growth. The value of ρ ranges from 0.95 to 0.98, while μ is fixed to be 0.5. The figure also shows the upper bound based on busy period measurements. This curve agrees with the formula in Equation (14) as expected.

B. The Lower Bound

Lemma 8: The expected per-packet delay is lower bounded by $\Omega\left(\frac{1}{1-\rho}\right)$

Proof: The expected per-packet delay for the single receiver case is clearly a lower bound for the corresponding

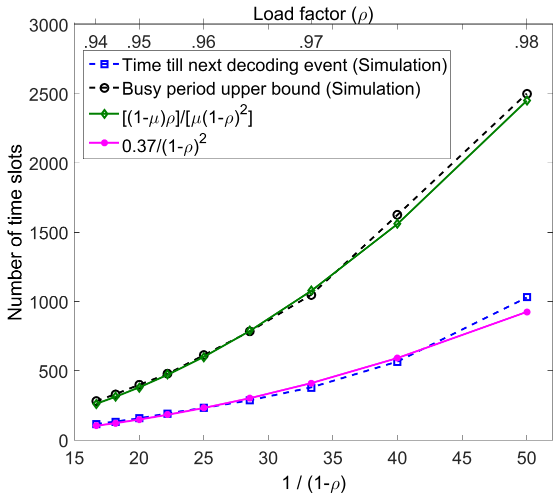

Fig. 4. Delay to decoding event and upper bound for 2 receiver case, as a function of $\frac{1}{(1-\rho)}$. The corresponding values of ρ are shown on the top of the figure.

quantity at one of the receivers in a multiple-receiver system. Figure 2 shows the Markov chain for the queue size in the single receiver case. If $\rho = \frac{\lambda}{\mu} < 1$, then the chain is positive recurrent and the steady state expected queue size can be computed to be $\frac{\rho(1-\mu)}{(1-\rho)} = \Theta\left(\frac{1}{1-\rho}\right)$ (see Equation (1)). 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 $\Theta\left(\frac{1}{1-\rho}\right)$. ■

We now present the new coding module for the general case of any number of receivers. First, we describe the main ideas behind the algorithm. Then, we present the detailed specification.

C. Intuitive Description

The intuition behind the algorithm is to first identify for each receiver, the oldest packet that it has not yet decoded, which we will call the *request* of that receiver. The algorithm then transmits a linear combination that involves packets from only within this set.

The linear combination is constructed incrementally. The receivers are grouped according to their request, and the groups are processed in descending order of their requested packet's index. First, the newest request (*i.e.*, the one with the largest index) is included in the linear combination, as otherwise, the corresponding receivers, having decoded everything older, will find the transmission non-innovative. Then, the algorithm checks whether the linear combination formed thus far is innovative to every receiver in the next group. If it is not innovative, then the coefficient of the next group's request is adjusted till it is simultaneously innovative to the whole group. The key idea is that, since the groups are processed in descending order of their requests, the choices made for the coefficient of subsequent groups' requests will not affect the innovation of earlier groups. This is because, the earlier groups have already decoded the subsequent groups' requests.

After processing all the groups in this order, the transmitted linear combination is thus chosen so that it satisfies the innovation guarantee property.

D. Representing Knowledge

Before specifying the algorithm, we first propose a way to systematically represent the state of knowledge of the receivers. This is based on the representation used in Section IV, with a key difference described below.

Let $\mathbf{p_k}$ denote the packet with index k. Suppose the total number of packets that have arrived at any time t is denoted by $A(t)$. Since the code is linear, we can represent the state of knowledge of a node by a vector space consisting of all the linear combinations that a node can compute using what it has received thus far. We represent the state of knowledge using a basis of this vector space. The basis is represented as the rows of a matrix which is in the reduced row echelon form (RREF). The matrix has $A(t)$ columns, one for each packet that has arrived thus far. While all this is identical to the representation in Section IV, the main difference is in the ordering of the columns of the basis matrix. We use the same framework, except that in this section, the columns of the RREF basis matrix are ordered so that packet $\mathbf{p_k}$ maps to column $A(t) - k$. In other words, the columns are arranged in reverse order with respect to the order of arrival at the sender.

Throughout this section, we will use the RREF representation of the basis matrix, with this reverse ordering of the packets. We also make use of the notion of seen packets that was introduced in Section II-A. Note however that the definition becomes quite different from the previous definition if we use the reverse ordering on the packets.

Definition 13 (Seeing a Packet With Reverse Ordering): A node is said to have *seen* a packet with index k if and only if the kth column from the right, of the RREF basis B of the knowledge space V of the node, is a pivot column. Alternatively, a node has seen a packet $\mathbf{p_k}$ if it has received enough information to compute a linear combination of the form $(\mathbf{p_k}+\mathbf{q})$, where $\mathbf{q}$ is itself a linear combination involving only packets with an index ***less*** than that of $\mathbf{p}$. (Decoding implies seeing, as we can pick $\mathbf{q} = \mathbf{0}$.)
In contrast, the definition used in Section II-A had replaced the word "less" with the word "greater" in the above statement. We believe the reverse ordering is better suited to analyzing the delivery delay. We now make some observations about the new definition.

Observation 1: As with the forward ordering, the notion of seen with reverse ordering also has connections to the dimension of the knowledge space. In particular, we can show that every innovative reception causes a new packet to be seen. In other words, the number of seen packets is equal to the dimension of the knowledge space.

Observation 2: Due to the reverse ordering of the packets, we have an interesting property. For any $k > 0$, if all packets $\mathbf{p_1}$ to $\mathbf{p_k}$ have been seen, then they have also been decoded, and hence can be delivered to the application.

E. Algorithm Specification

Now, we present the formal coding algorithm. Note that the algorithm and its analysis use the notion of seen packets with reverse ordering.

Let $\{u_1, u_2, \ldots, u_m\}$ be the set of indices of the oldest undecoded packets of the n receivers, sorted in descending order ($m \leq n$, since the oldest undecoded packet may be the same for some receivers). Exclude receivers whose oldest undecoded packet has not yet arrived at the sender. We call this resulting set of packets the *transmit set*, since the coding module will use only these packets in computing the linear combination to be transmitted.

Let $R(u_i)$ be the group of receivers whose request is $\mathbf{p_{u_i}}$. We now present the coding module to select the linear combination for transmission.

```
Initialize the transmit coefficient vector a to an all zero
vector of length $Q$, the current sender queue size.
for $j = 1$ to $m$ do {    (Loop over the transmit set)}
  Initialize the veto list[6] to the empty set.
  for all $r \in R(u_j)$ do
    Zero out the coefficient of all packets seen by receiver $r$
    from the current transmission vector a by subtracting
    from a, suitably scaled versions of the rows of the
    current RREF basis matrix, to get the vector a'. (This
    is essentially the first step of Gauss-Jordan elimination.)
    Hence, find out which packet will be newly seen if the
    linear combination corresponding to a is transmitted.
    This is simply the index of the packet corresponding
    to the first non-zero entry in a'.
    if no packet is newly seen then
      Append 0 to the veto list
    else if the newly seen packet's index is $u_j$ then
      Append the additive inverse of the leading non-zero
      entry of a' to the veto list
    else if the newly seen packet is anything else then
      Do not add anything to the veto list
    end if
  end for
  Arrange the elements of the finite field in any order,
  starting with 0. Choose $a_{u_j}$ to be the first element in this
  order that is not in the veto list.
end for
Compute the transmit packet:   $\mathbf{g} := \sum_{k=1}^{Q} a_k \mathbf{p_k}$
```

F. Properties of the Algorithm

1) Throughput: To ensure correctness, the algorithm uses a finite field of size at least as large as the number of receivers. Theorem 11 shows that this is a sufficient condition to guarantee innovation and thereby achieve optimal throughput.

Theorem 11: If the field is at least as large as the number of receivers, then the above algorithm will always find values for the a_k's such that the resulting transmission satisfies the innovation guarantee property.

Proof: We first show that the choices made by the algorithm guarantee innovation. For any $j > 0$, consider the jth request group. Let $\mathbf{a}(j-1)$ be the value of the coefficient vector just before processing group j (Note, $\mathbf{a(0)} = \mathbf{0}$.).

Any receiver in group j has not decoded $\mathbf{p_{u_j}}$ yet. Hence, it cannot know a linear combination of the form $\mathbf{a}(j-1)+\beta\mathbf{e_{u_j}}$

[6]This will hold the list of unacceptable coefficients of $\mathbf{p_{u_j}}$.

for more than one value of β, where $\mathbf{e_{u_j}}$ is the unit vector with a 1 in the u_jth coordinate and 0 elsewhere. (If it knew two such combinations, it could subtract one from the other to find $\mathbf{p_{u_j}}$, a contradiction.)

Suppose the receiver knows exactly one such linear combination. Then, after the row reduction step, the vector $\mathbf{a}(j-1)$ will get transformed into $\mathbf{a}' = -\beta\mathbf{e_{u_j}}$. Hence, the leading non-zero coefficient of $\mathbf{a}'$ is $-\beta$, and its additive inverse gives β. (Note: the resulting value of β could be 0. This corresponds to the non-innovative case.) If the receiver does not know any linear combination of this form, then packet u_j is not seen, and nothing is added to the veto list.

In short, the values that are vetoed are those values of β for which some receiver knows a linear combination of the form $\mathbf{a}(j-1)+\beta\mathbf{e_{u_j}}$. Hence, by picking a value of a_{u_j} from outside this list, we ensure innovation. Thus, the algorithm essentially checks for innovation by considering different coefficients β for including $\mathbf{p_{u_j}}$ into the transmission and eliminating the ones that do not work. Finally, processing subsequent groups will not affect the innovation of the previous groups because the subsequent groups will only change the coefficient of their requests, which have already been decoded by the previous groups.

We now show that the algorithm always has enough choices to pick such an a_{u_j} even after excluding the veto list. As argued above, at any point in the algorithm, each receiver adds at most one field element to the veto list. Hence, the veto list can never be longer than the number of receivers in the corresponding request group. Now, we consider two cases.

Case 1: If the group requesting the highest request u_1 does not include all the receivers, then none of the groups contain n receivers. Hence, the veto list for any group will always be strictly shorter than n, and hence if the field size is at least n, there is always a choice left for a_{u_j}.

Case 2: If all n receivers request the highest packet u_1, then it has to be the case that they have all decoded every packet before u_1. Hence, the only coefficient that any receiver would veto for $\mathbf{p_{u_1}}$ is 0, thus leaving other choices for a_{u_1}.

This completes the proof. ■

2) Decoding and Delivery Delay: We conjecture that the coding module described above has good delay performance.

Conjecture 1: For the coding module in Section VI-E, the expected decoding delay per packet, as well as the expected delivery delay per packet with respect to a particular receiver, grow as $O\left(\frac{1}{1-\rho}\right)$ *as* $\rho \to 1$, *which is asymptotically optimal.*

The exact analysis of the delay and the proof of this conjecture are open problems. We believe that the notion of seen packets will be useful in this analysis. In particular, to analyze the delivery delay, we can make use of Observation 2 from Section VI-D. A packet is delivered if and only if this packet and all packets with a lower index have been seen. This condition is the same as what arises in problems involving a resequencing buffer. Thus, we can formulate our delivery delay problem in terms of traditional queuing problems.

In our formulation, we break down the delivery delay of a packet for a particular receiver into two parts, as though the packet has to traverse two queues in tandem. The first part is simply the time until the packet is seen. Once it is seen, the packet moves into a second queue which is essentially a resequencing buffer. The second part is the time spent in this buffer waiting for all older packets to be seen.

The expectation of the first part is easy to calculate, since every innovative reception causes a new packet to be seen. By Little's theorem, the delay is directly proportional to the size of the queue of unseen packets. This queue's behavior was studied in Section IV. Although that section used the older notion of seeing a packet, it can be shown that the analysis still holds even if we use the new notion of seen packets based on reverse ordering. Hence, we get a $O\left(\frac{1}{1-\rho}\right)$ bound on the first part of the delay. The analysis of the second part of the delay however, seems more complicated.

3) Queue Management: The coding module described above makes use of only the oldest undecoded packet of each receiver in any given time-slot. Since our definition of seen packets uses reverse ordering of the packets (see Section VI-D), the oldest undecoded packet is always an unseen packet. In other words, the algorithm never uses packets that have been seen by all the receivers. This implies that the algorithm is compatible with the drop-when-seen queuing algorithm (Algorithm 2 (b))that was proposed and analyzed in Section IV, provided we use the new definition of seen. As pointed out in Observation 1 in Section VI-D, the new definition of seeing a packet has the same relation to the dimension of the knowledge space as the old definition of Section VI-D. Thus, we can obtain all the queue size guarantees that were obtained in the earlier work. In other words, we can get a provable $O\left(\frac{1}{1-\rho}\right)$ growth of the expected queue size at the sender, in addition to the provable innovation guarantee property and the conjectured delay guarantees.

G. Simulation Results

We now evaluate the performance of the newly proposed coding module through simulations. In particular, we study the behavior of the decoding delay and the delivery delay as a function of the load factor ρ, in the limit as ρ approaches 1, *i.e.*, as the loading on the system approaches capacity.

The probability of reception in any slot is $\mu = 0.5$. The packets arrive according to a Bernoulli process, whose arrival rate is calculated according to the load factor ρ. The load factor is varied through the following values: 0.8, 0.9, 0.92, 0.94, 0.96, 0.97, 0.98 and 0.99. The decoding delay and delivery delay are averaged across the packets over a large number of slots. The number of slots is set to 10^6 for the first four data points, 2×10^6 for the next two points, and at 5×10^6 for the last two points.

We consider two different cases. In the first case, there are three receivers. The entire operation is therefore performed over a $GF(3)$ (*i.e.*, integer operations modulo 3). In the second case, we consider the situation where there are five receivers. In this case, the operations are performed over a field of size 5.

Figure 5 shows the plot of the decoding and delivery delay as a function of $\frac{1}{1-\rho}$ for both the three and the five receiver cases. Figure 6 shows the same plot in a logarithmic scale. From both these figures, it is clearly seen that the algorithm

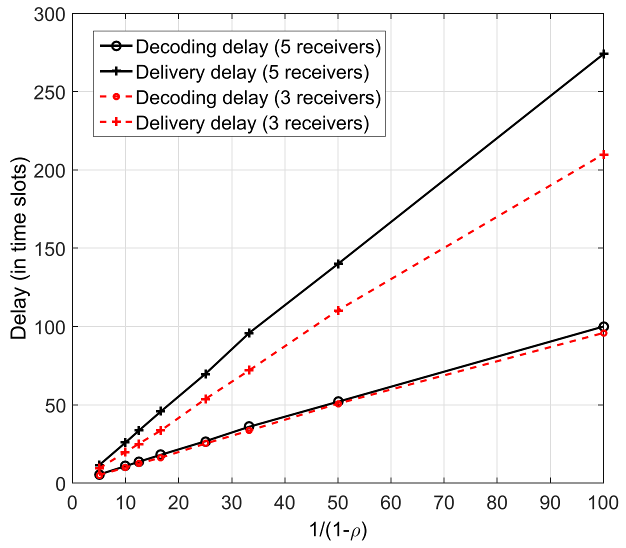

Fig. 5. Linear plot of the decoding and delivery delay.

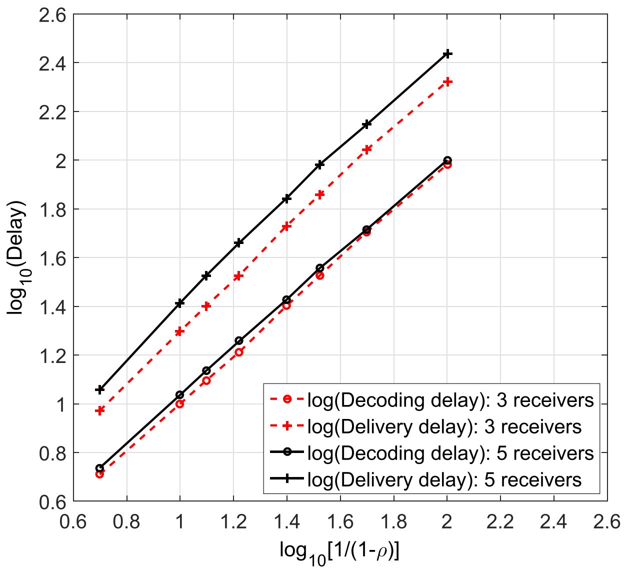

Fig. 6. Log plot of the decoding and delivery delay.

achieves a linear growth of the delay in terms of $\frac{1}{1-\rho}$. We have thus verified Conjecture 1 for the case of 3 and 5 receivers, using simulations.

VII. Applications and Further Extensions

Although we have presented the algorithms in the context of a single packet erasure broadcast channel, we believe the main ideas in the scheme are quite robust and can be applied to more general topologies. The scheme readily extends to a tandem network of broadcast links (with no mergers) if the intermediate nodes use the witness packets in place of the original packets. The notion of encoding only unseen packets, acknowledging seen packets, and the algorithm of dropping packets that each receiver has acknowledged to have seen are applicable even at intermediate hops in a multi-hop network coding system. The format for conveying the coding vectors across multiple hops will need careful design. We expect that it will also extend to other topologies with suitable modifications. In addition, we believe the proposed scheme will also be robust to delayed or imperfect feedback, just like conventional ARQ. Such a generalization can lead to a TCP-like protocol for systems that use network coding [36].

We have assumed the erasures to be independent and identically distributed across receivers. However, the analysis for Algorithm 2 (b) will hold even if we allow adversarial erasures. This is because, the guarantee that the physical queue size tracks the backlog in degrees of freedom is not a probabilistic guarantee, but a combinatorial guarantee on the instantaneous value of the queue sizes. Note that, while the erasures can be chosen adversarially, we will require the adversary to guarantee a certain minimum long-term connection rate from the sender to every receiver, so that the virtual queues can themselves be stabilized.

From a theoretical point of view, our results mean that any stability results or queue size bounds in terms of virtual queues can be translated to corresponding results for the physical queues. In addition, results from traditional queuing theory about M/G/1 queues or a Jackson network type of result [8] can be extended to the physical queue size in coded networks, as opposed to just the backlog in degrees of freedom. From a practical point of view, if the memory at the sender has to be shared among several different flows, then this reduction in queue occupancy will prove quite useful in getting statistical multiplexing benefits.

For instance, one specific scenario where our results can be immediately applied is the multicast switch with intra-flow network coding, studied in [20]. The multicast switch has broadcast-mode links from each input to all the outputs. “Erasures” occur because the scheduler may require that only some outputs can receive the transmission, as the others are scheduled to receive a different transmission from some other input. In this case, there is no need for explicit feedback, since the sender can track the states of knowledge of the receivers simply using the scheduling configurations from the past. The results stated in [20] in terms of the virtual queues can thus be extended to the physical queues as well.

VIII. Conclusion

In this work, we have presented a completely online approach to network coding based on feedback, which does not compromise on throughput and yet, provides benefits in terms of queue occupancy at the sender and decoding and delivery delay at the receivers.

The notion of seen packets introduced in this work, allows the application of tools and results from traditional queuing theory in contexts that involve coding across packets. Using this notion, we proposed the drop-when-seen algorithm, which allows the physical queue size to track the backlog in degrees of freedom, thereby reducing the amount of storage used at the sender. Comparing the results in Theorem 1 and Theorem 6, we see that the newly proposed Algorithm 2 (b) gives a significant improvement in the expected queue size at the sender, compared to Algorithm 1.

We have proposed a new coding scheme that makes use of feedback to dynamically adapt the code in order to ensure low decoding delay for any number of receivers. As argued earlier, $\Theta\left(\frac{1}{1-\rho}\right)$ is an asymptotic lower bound on the decoding delay and the stronger notion of delivery delay in the limit of the load factor approaching capacity ($\rho \to 1$). We conjecture that our scheme achieves this lower bound. If true, this implies the asymptotic optimality of our coding module in terms of both decoding delay and delivery delay. We have verified this conjecture through simulations.

In summary, we believe that the proper combination of feedback and coding in erasure networks presents a wide range of benefits in terms of throughput, queue management and delay. Our work is a step towards realizing these benefits.

Appendix A
Proof of Theorem 1

Proof: Let T be the time an arbitrary arrival in steady state spends in the physical queue before departure, excluding the slot in which the arrival occurs (Thus, if a packet departs immediately after it arrives, then T is 0.). A packet in the physical queue will depart when each virtual queue has become empty at least once since its arrival. Let D_j be the time starting from the new arrival, until the next emptying of the jth virtual queue. Then, $T = \max_j D_j$ and so, $\mathbb{E}[T] \geq \mathbb{E}[D_j]$. Hence, we focus on $\mathbb{E}[D_j]$.

We condition on the event that the state seen by the new arrival just before it joins the queue, is some state k. There are two possibilities for the queue state at the end of the slot in which the packet arrives. If the channel is ON in that slot, then there is a departure and the state at the end of the slot is k. If the channel is OFF, then there is no departure and the state is $(k+1)$. Now, D_j is simply the first passage time from the state at the end of that slot to state 0, *i.e.*, the number of slots it takes for the system to reach state 0 for the first time, starting from the state at the end of the arrival slot. Let $\Gamma_{u,v}$ denote the expected first passage time from state u to state v. The expected first passage time from state u to state 0, for $u > 0$ is derived in Appendix B, and is given by the following expression:

$$\Gamma_{u,0} = u/(\mu - \lambda)$$

Now, because of the property that Bernoulli arrivals see time averages (BASTA) [37], an arbitrary arrival sees the same distribution for the size of the virtual queues, as the steady state distribution given in Equation (1).

Using this fact, we can compute the expectation of D_j as follows:

$$\begin{aligned}\mathbb{E}[D_j] &= \sum_{k=0}^{\infty} \mathbb{P}(\text{New arrival sees state } k)\mathbb{E}[D_j|\text{State } k] \\ &= \sum_{k=0}^{\infty} \pi_k[\mu\Gamma_{k,0} + (1-\mu)\Gamma_{k+1,0}] \\ &= \sum_{k=0}^{\infty} \pi_k \cdot \frac{\mu k + (1-\mu)(k+1)}{\mu - \lambda} \\ &= \frac{1-\mu}{\mu} \cdot \frac{\rho}{(1-\rho)^2} \end{aligned} \tag{14}$$

Now, the expected time that an arbitrary arrival in steady state spends in the system is given by:

$$\mathbb{E}[T] = \mathbb{E}[\max_j D_j] \geq \mathbb{E}[D_j] = \Omega\left(\frac{1}{(1-\rho)^2}\right)$$

Since each virtual queue is positive recurrent (assuming $\lambda < \mu$), the physical queue will also become empty infinitely often. Then we can use Little's law to find the expected physical queue size.

The expected size of the physical queue in steady state if we use algorithm 1 is given by:

$$\lim_{t\to\infty} \mathbb{E}[Q(t)] = \lambda\mathbb{E}[T] = \Omega\left(\frac{1}{(1-\rho)^2}\right)$$

■

Appendix B
Derivation of the First Passage Time

Consider the Markov chain $\{Q_j(t)\}$ for the virtual queue size, shown in Figure 2. Assume that the Markov chain has an initial distribution equal to the steady state distribution (Equivalently, assume that the Markov chain has reached steady state.). We use the same notation as in Section IV-A.

Define $N_m := \inf\{t \geq 1 : Q_j(t) = m\}$. We are interested in deriving for $k \geq 1$, an expression for $\Gamma_{k,0}$, the expected first passage time from state k to 0, *i.e.*,

$$\Gamma_{k,0} = \mathbb{E}[N_0|Q_j(0) = k]$$

Define for $i \geq 1$: $X_i := a(i) - d(i)$,
where $a(i)$ is the indicator function for an arrival in slot i, and $d(i)$ is the indicator function for the channel being on in slot i. Let $S_t := \sum_{i=1}^{t} X_i$. If $Q_j(t) > 0$, then the channel being on in slot t implies that there is a departure in that slot. Thus the correspondence between the channel being on and a departure holds for all $0 \leq t \leq N_0$. This implies that:

$$\text{For } t \leq N_0,\ Q_j(t) = Q_j(0) + S_t$$

Thus, N_0 can be redefined as the smallest $t \geq 1$ such that S_t reaches $-Q_j(0)$. Thus, N_0 is a valid stopping rule for the X_i's which are themselves IID, and have a mean $\mathbb{E}[X] = (\lambda - \mu)$. We can find $\mathbb{E}[N_0]$ using Wald's equality:

$$\begin{aligned}\mathbb{E}[S_{N_0}|Q_j(0) = k] &= \mathbb{E}[N_0|Q_j(0) = k] \cdot \mathbb{E}[X] \\ \text{i.e., } -k &= \mathbb{E}[N_0|Q_j(0) = k] \cdot (\lambda - \mu) \\ \Rightarrow \Gamma_{k,0} &= \mathbb{E}[N_0|Q_j(0) = k] = \frac{k}{\mu - \lambda}\end{aligned}$$

Appendix C
Proof of Lemma 1

Proof: We show that any completion of B_Δ into a basis of $span(B_j)$ can be changed to a basis with the required property.

Let $B_\Delta = \{\mathbf{b_1}, \mathbf{b_2}, \ldots, \mathbf{b_m}\}$. Suppose we complete this into a basis C_j of $span(B_j)$ such that:

$$C_j = B_\Delta \cup \{\mathbf{c_1}, \mathbf{c_2}, \ldots, \mathbf{c_{|B_j|-m}}\}$$

Now, we claim that at the beginning of step 6, $span(B_j) \subseteq span(B)$ for all j. This can be proved by induction on the slot number, using the way the algorithm updates B and the B_j's. Intuitively, it says that any receiver knows a subset of what the sender knows.

Therefore, for each vector $\mathbf{c} \in C_j \backslash B_\Delta$, $\mathbf{c}$ must also be in $span(B)$. Now, since $B_\Delta \cup B''$ is a basis of $span(B)$, we can write $\mathbf{c}$ as $\sum_{i=1}^{m} \alpha_i \mathbf{b_i} + \mathbf{c}'$ with $\mathbf{c}' \in span(B'')$. In this manner, each $\mathbf{c_i}$ gives a distinct $\mathbf{c_i'}$. It is easily seen that

$C'_j := B_\Delta \cup \{\mathbf{c'_1}, \mathbf{c'_2}, \ldots, \mathbf{c'_{|B_j|-m}}\}$ is also a basis of the same space that is spanned by C_j. Moreover, it satisfies the property that $C'_j \backslash B_\Delta \subseteq span(B'')$. ■

Appendix D
Proof of Lemma 3

Proof: The proof is by induction on t.

Basis step: In the beginning of slot 1, $a(1)$ packets arrive. So, $H(1) = I_{a(1)}$ and hence the rows are linearly independent.

Induction hypothesis: Assume $H(t-1)$ has linearly independent rows.

Induction step: The queue is updated such that the linear combinations corresponding to local coefficient vectors in B'' are stored, and subsequently, the $a(t)$ new arrivals are appended. Thus, the relation between $H(t-1)$ and $H(t)$ is:

$$H(t) = \begin{bmatrix} B''(t-1)H(t-1) & 0 \\ 0 & I_{a(t)} \end{bmatrix}$$

Now, $B''(t-1)$ has linearly independent rows, since the rows form a basis. The rows of $H(t-1)$ are also linearly independent by hypothesis. Hence, the rows of $B''(t-1)$ $H(t-1)$ will also be linearly independent. Appending $a(t)$ zeros and then adding an identity matrix block in the right bottom corner does not affect the linear independence. Hence, $H(t)$ also has linearly independent rows. ■

Appendix E
Proof of Lemma 4

Proof: For any $z \in V_\Delta \oplus \cap_{i=1}^n U_i$, there is a $x \in V_\Delta$ and $y \in \cap_{i=1}^n U_i$ such that $z = x + y$. Now, for each i, $y \in U_i$. Thus, $z = x + y$ implies that $z \in \cap_{i=1}^n [V_\Delta \oplus U_i]$. Therefore, $V_\Delta \oplus \cap_{i=1}^n U_i \subseteq \cap_{i=1}^n [V_\Delta \oplus U_i]$.

Now, let $w \in \cap_{i=1}^n V_\Delta \oplus U_i$. Then for each i, there is a $x_i \in V_\Delta$ and $y_i \in U_i$ such that $w = x_i + y_i$. But, $w = x_i + y_i = x_j + y_j$ means that $x_i - x_j = y_i - y_j$. Now, $(x_i - x_j) \in V_\Delta$ and $(y_i - y_j) \in (U_1 + U_2 + \ldots + U_n)$. By hypothesis, these two vector spaces have only 0 in common. Thus, $x_i - x_j = y_i - y_j = 0$. All the x_i's are equal to a common $x \in V_\Delta$ and all the y_i's are equal to a common y which belongs to all the U_i's. This means, w can be written as the sum of a vector in V_Δ and a vector in $\cap_{i=1}^n U_i$, thereby proving that $\cap_{i=1}^n [V_\Delta \oplus U_i] \subseteq V_\Delta \oplus \cap_{i=1}^n U_i$. ■

Appendix F
Proof of Lemma 5

Proof: Statement 1 follows from the fact that B is a subset of $B \oplus C$. Hence, if $A \cap (B \oplus C)$ is empty, so is $A \cap B$.

For statement 2, we need to show that $(A \oplus B) \cap C = \{\mathbf{0}\}$. Consider any element $\mathbf{x} \in (A \oplus B) \cap C$. Since it is in $A \oplus B$, there exist unique $\mathbf{a} \in A$ and $\mathbf{b} \in B$ such that $\mathbf{x} = \mathbf{a} + \mathbf{b}$. Now, since $\mathbf{b} \in B$ and $\mathbf{x} \in C$, it follows that $\mathbf{a} = \mathbf{x} - \mathbf{c}$ is in $B \oplus C$. It is also in A. Since A is independent of $B \oplus C$, $\mathbf{a}$ must be $\mathbf{0}$. Hence, $\mathbf{x} = \mathbf{b}$. But this means $\mathbf{x} \in B$. Since it is also in C, it must be $\mathbf{0}$, as B and C are independent. This shows that the only element in $(A \oplus B) \oplus C$ is $\mathbf{0}$.

Statement 3 can be proved as follows.

$$\begin{aligned}
&\mathbf{x} \in A \oplus (B \oplus C) \\
&\quad \Leftrightarrow \exists \text{ unique } \mathbf{a} \in A, \mathbf{d} \in B \oplus C \text{ s.t. } \mathbf{x} = \mathbf{a} + \mathbf{d} \\
&\quad \Leftrightarrow \exists \text{ unique } \mathbf{a} \in A, \mathbf{b} \in B, \mathbf{c} \in C \text{ s.t. } \mathbf{x} = \mathbf{a} + \mathbf{b} + \mathbf{c} \\
&\quad \Leftrightarrow \exists \text{ unique } \mathbf{e} \in A \oplus B, \mathbf{c} \in C \text{ s.t. } \mathbf{x} = \mathbf{e} + \mathbf{c} \\
&\quad \Leftrightarrow \mathbf{x} \in (A \oplus B) \oplus C
\end{aligned}$$

■

Appendix G
Proof of Lemma 6

Proof: For any two subspaces X and Y of V, let $X + Y$ denotes the span of subspaces X and Y. Then,

$$\begin{aligned}
dim(X \cap Y) &= dim(X) + dim(Y) - dim(X + Y) \\
&\geq dim(X) + dim(Y) - dim(V) \\
&\quad \text{(since } X + Y \text{ is also a subspace of } V\text{)} \quad (15)
\end{aligned}$$

Now, we prove the lemma by induction on k.

Basis step:

$k = 1$: LHS $= dim(V_1)$, RHS $= dim(V_1)$

$k = 2$: LHS $= dim(V_1 \cap V_2)$, RHS $= dim(V_1) + dim(V_2) - dim(V)$

The claim follows from inequality (15).

Induction Hypothesis:

For some arbitrary k,

$$dim(\cap_{i=1}^{k-1} V_i) \geq \sum_{i=1}^{k-1} dim(V_i) - (k-2)dim(V)$$

Induction Step:

$$\begin{aligned}
&dim(\cap_{i=1}^{k} V_i) \\
&= dim(V_k \cap \cap_{i=1}^{k-1} V_i) \\
&\geq dim(V_k) + dim(\cap_{i=1}^{k-1} V_i) - dim(V) \quad \text{(using (15))} \\
&\geq dim(V_k) + \left[\sum_{i=1}^{k-1} dim(V_i) - (k-2)dim(V)\right] \\
&\quad - dim(V) \\
&= \sum_{i=1}^{k} dim(V_i) - (k-1)dim(V)
\end{aligned}$$

■

References

[1] M. Luby, "LT codes," in *Proc. 43rd Annu. IEEE Symp. Found. Comput. Sci.*, Nov. 2002, pp. 271–282.

[2] A. Shokrollahi, "Raptor codes," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jul. 2004, p. 36.

[3] P. Pakzad, C. Fragouli, and A. Shokrollahi, "Coding schemes for line networks," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Sep. 2005, pp. 1853–1857.

[4] R. Gummadi and R. S. Sreenivas, "Relaying a fountain code across multiple nodes," in *Proc. IEEE Inf. Theory Workshop*, May 2008, pp. 149–153.

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

[6] R. Koetter and M. Médard, "An algebraic approach to network coding," *IEEE/ACM Trans. Netw.*, vol. 11, no. 5, pp. 782–795, Oct. 2003.

[7] D. S. Lun, "Efficient operation of coded packet networks," Ph.D. dissertation, Dept. EECS, Massachusetts Inst. Technol., Cambridge, MA, USA, Jun. 2006.

[8] D. S. Lun, M. Médard, R. Koetter, and M. Effros, "On coding for reliable communication over packet networks," *Phys. Commun.*, vol. 1, no. 1, pp. 3–20, 2008.

[9] E. Martinian, "Dynamic information and constraints in source and channel coding," Ph.D. dissertation, Dept. EECS, Massachusetts Inst. Technol., Cambridge, MA, USA, Sep. 2004.

[10] A. Sahai, "Why delay and block length are not the same thing for channel coding with feedback," in *Proc. UCSD Workshop Inf. Theory Appl.*, Feb. 2006, p. 26.

[11] C. T. K. Ng, M. Médard, and A. Ozdaglar, "Completion time minimization and robust power control in wireless packet networks," in *Proc. IEEE Int. Conf. Commun. (ICC)*, Dresden, Germany, Jun. 2009, pp. 1–6.

[12] S. Sanghavi, "Intermediate performance of rateless codes," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Sep. 2007, pp. 478–482.

[13] A. Beimel, S. Dolev, and N. Singer, "RT oblivious erasure correcting," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Oct. 2004, pp. 236–241.

[14] A. Albanese, J. Blomer, J. Edmonds, M. Luby, and M. Sudan, "Priority encoding transmission," *IEEE Trans. Inf. Theory*, vol. 42, no. 6, pp. 1737–1744, Nov. 1996.

[15] D. Silva and F. R. Kschischang, "Rank-metric codes for priority encoding transmission with network coding," in *Proc. 10th Can. Workshop Inf. Theory*, Jun. 2007, pp. 81–84.

[16] J. M. Walsh and S. Weber, "A concatenated network coding scheme for multimedia transmission," in *Proc. NetCod*, Jan. 2008, pp. 1–6.

[17] T. Ho and H. Viswanathan, "Dynamic algorithms for multicast with intra-session network coding," in *Proc. 43rd Allerton Annu. Conf. Commun. Control Comput.*, Monticello, IL, USA, 2005, pp. 2063–2072.

[18] A. Eryilmaz and D. S. Lun, "Control for inter-session network coding," in *Proc. NetCod*, Jan. 2007, pp. 1–9.

[19] M. Artin, *Algebra*. Englewood Cliffs, NJ, USA: Prentice-Hall, 1991.

[20] J. K. Sundararajan, M. Médard, M. Kim, A. Eryilmaz, D. Shah, and R. Koetter, "Network coding in a multicast switch," in *Proc. IEEE INFOCOM*, May 2007, pp. 1145–1153.

[21] C. Fragouli, D. S. Lun, M. Médard, and P. Pakzad, "On feedback for network coding," in *Proc. 41st Annu. Conf. Inf. Sci. Syst. (CISS)*, Mar. 2007, pp. 248–252.

[22] B. Shrader and A. Ephremides, "On the queueing delay of a multicast erasure channel," in *Proc. IEEE Inf. Theory Workshop (ITW)*, Oct. 2006, pp. 423–427.

[23] B. Shrader and A. Ephremides, "A queueing model for random linear coding," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Oct. 2007, pp. 1–7.

[24] Y. E. Sagduyu and A. Ephremides, "On broadcast stability region in random access through network coding," in *Proc. 44th Allerton Annu. Conf. Commun. Control Comput.*, Sep. 2006, pp. 143–150.

[25] Y. E. Sagduyu and A. Ephremides, "On network coding for stable multicast communication," in *Proc. IEEE Military Commun. Conf. (MILCOM)*, Oct. 2007, pp. 1–7.

[26] J. Lacan and E. Lochin, "On-the-fly coding to enable full reliability without retransmission," ISAE, LAAS-CNRS, Toulouse, France, Tech. Rep., 2008. [Online]. Available: http://arxiv.org/pdf/0809.4576

[27] P. Larsson and N. Johansson, "Multi-user ARQ," in *Proc. IEEE 63rd Veh. Technol. Conf. (VTC)*, May 2006, pp. 2052–2057.

[28] K. Jolfaei, S. Martin, and J. Mattfeldt, "A new efficient selective repeat protocol for point-to-multipoint communication," in *Proc. IEEE Int. Conf. Commun. (ICC)*, vol. 2. Geneva, Switzerland, May 1993, pp. 1113–1117.

[29] S. Yong and L. B. Sung, "XOR retransmission in multicast error recovery," in *Proc. IEEE Int. Conf. Netw. (ICON)*, Sep. 2000, pp. 336–340.

[30] P. Larsson, "Multicast multiuser ARQ," in *Proc. IEEE Wireless Commun. Netw. Conf. (WCNC)*, Apr. 2008, pp. 1985–1990.

[31] D. Nguyen, T. Tran, T. Nguyen, and B. Bose, "Wireless broadcast using network coding," *IEEE Trans. Veh. Technol.*, vol. 58, no. 2, pp. 914–925, Feb. 2009.

[32] M. Durvy, C. Fragouli, and P. Thiran, "Towards reliable broadcasting using ACKs," in *Proc. IEEE Int. Symp. Inf. Theory (ISIT)*, Jun. 2007, pp. 1156–1160.

[33] L. Keller, E. Drinea, and C. Fragouli, "Online broadcasting with network coding," in *Proc. NetCod*, Jan. 2008, pp. 1–6.

[34] J. Barros, R. A. Costa, D. Munaretto, and J. Widmer, "Effective delay control in online network coding," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 208–216.

[35] J. J. Hunter, *Mathematical Techniques of Applied Probability, Discrete Time Models: Techniques and Applications*, vol. 2. New York, NY, USA: Academic, 1983.

[36] J. K. Sundararajan, D. Shah, M. Médard, M. Mitzenmacher, and J. Barros, "Network coding meets TCP," in *Proc. IEEE INFOCOM*, Apr. 2009, pp. 280–288.

[37] H. Takagi, *Queueing Analysis: Discrete-Time Systems*, vol. 3. Amsterdam, The Netherlands: Elsevier, 1993.

Jay Kumar Sundararajan (S'02–M'10) is a systems engineer at Qualcomm Research, San Diego, CA, USA. He received the B.Tech. degree from the Department of Electrical Engineering, Indian Institute of Technology Madras, India in 2003 and the SM and PhD degrees from the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA in 2005 and 2009 respectively. His research interests are in the area of network coding, scheduling and wireless communications. He received the 2008 Marconi Young Scholar Award in recognition of his graduate research.

Devavrat Shah (M'05–SM'16) is a Professor with the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. His current research interests are at the interface of Statistical Inference and Social Data Processing. His work has been recognized through prize paper awards in Machine Learning, Operations Research and Computer Science, as well as career prizes including 2010 Erlang prize from the INFORMS Applied Probability Society and 2008 ACM Sigmetrics Rising Star Award. He is a distinguished young alumnus of his alma mater, Indian Institute of Technology Bombay. He has served as an Associate Editor for IEEE Transactions on Information Theory ('12–'16), Queuing Systems ('12–'16) and IEEE Transactions on Network Science and Engineering ('14–'17).

Muriel Médard (S'91–M'95–SM'02–F'08) is the Cecil H. Green Professor of Electrical Engineering and Computer Science at the Department of Electrical Engineering and Computer Science, Massachusetts Institute of Technology (MIT), Cambridge, MA, USA. Her research interests are in the areas of network coding and reliable communications. She has served as an Editor of many IEEE publications, and is currently the Editor-in-Chief of the IEEE JOURNAL ON SELECTED AREAS IN COMMUNICATIONS. She served on the Board of Governors of the IEEE Information Theory Society, for which she was President in 2012. She has served as a TPC Chair or general Chair for several IEEE conferences. She was the recipient of the 2013 MIT Graduate Student Council EECS Mentor Award, the 2009 Communication Society and Information Theory Society Joint Paper Award, the 2009 William R. Bennett Prize in the Field of Communications Networking, the 2002 IEEE Leon K. Kirchmayer Prize Paper Award, and several conference paper awards. She was also a co-recipient of the MIT 2004 Harold E. Edgerton Faculty Achievement Award. In 2007, she was named a Gilbreth Lecturer by the U.S. National Academy of Engineering.

Parastoo Sadeghi (S'02–M'06–SM'07) is an Associate Professor at the Research School of Engineering, Australian National University, Canberra, Australia. She received the BSc and MSc degrees in electrical engineering from Sharif University of Technology, Tehran, Iran, in 1995 and 1997, respectively, and the PhD degree in electrical engineering from the University of New South Wales, Sydney, Australia, in 2006. From 1997 to 2002, she was a Research Engineer and then a Senior Research Engineer at Iran Communication Industries, Tehran, and at Deqx (formerly known as Clarity Eq), Sydney. She has visited various research institutes, including the Institute for Communications Engineering, Technical University of Munich, in 2008 and MIT in 2009 and 2013.

Dr. Sadeghi has co-authored around 150 refereed journal or conference papers and a book on Hilbert Space Methods in Signal Processing (Cambridge Univ. Press, 2013). She is currently serving as an Associate Editor of the IEEE TRANSACTIONS ON INFORMATION THEORY. Dr. Sadeghi has been a Chief Investigator in a number of Australian Research Council Discovery and Linkage Projects. Her research interests are mainly in the areas of network coding, information theory, wireless communications theory, and signal processing.