entropy

MDPI

Article

On NACK-Based rDWS Algorithm for Network Coded Broadcast

Sovanjyoti Giri * and Rajarshi Roy

Department of Electronics and Electrical Communication Engineering, Indian Institute of Technology, Kharagpur, West Bengal 721302, India; royr@ece.iitkgp.ernet.in

* Correspondence: sgiri1990@iitkgp.ac.in

Received: 20 July 2019; Accepted: 27 August 2019; Published: 17 September 2019



Abstract: The *Drop when seen* (DWS) technique, an online network coding strategy is capable of making a broadcast transmission over erasure channels more robust. This throughput optimal strategy reduces the expected sender queue length. One major issue with the DWS technique is the high computational complexity. In this paper, we present a randomized version of the DWS technique (rDWS), where the unique strength of the DWS, which is the sender's ability to drop a packet even before its decoding at receivers, is not compromised. Computational complexity of the algorithms is reduced with rDWS, but the encoding is not throughput optimal here. So, we perform a throughput efficiency analysis of it. Exact probabilistic analysis of *innovativeness* of a coefficient is found to be difficult. Hence, we carry out two individual analyses, *maximum entropy* analysis, *average understanding* analysis, and obtain a lower bound on the *innovativeness* probability of a coefficient. Based on these findings, *innovativeness* probability of a coded combination is analyzed. We evaluate the performance of our proposed scheme in terms of dropping and decoding statistics through simulation. Our analysis, supported by plots, reveals some interesting facts about *innovativeness* and shows that rDWS technique achieves near-optimal performance for a finite field of sufficient size.

Keywords: broadcast; drop when seen; feedback; integer partition; online network coding; probability of innovativeness; randomized drop when seen

1. Introduction

Reliable communication through packet erasure channels is one of the key challenges of modern wireless systems. Digital fountain codes [1–3] are a class of codes that can be used as a tool to ensure reliability in such scenario. Though originally network codes were invented for achieving multicast capacity of a multicast network which consists of some fixed capacity faultless links, it can also be used to deal with heterogeneous packet-erasure networks. Network coding is preferred over fountain codes in some cases as decoding is not needed in the intermediate nodes, and it is more composable across links over various network configurations.

The idea of network coding was proposed by Ahlswade et al. [4] in the context of network information flow where information is not just a commodity like water which is flowing through a pipeline network. Breaking the traditional convention of routing, incoming information packets to a node are coded and forwarded to the next nodes across outgoing links. The main goal of this code and forward technique is to increase the throughput of the network, which may seem surprising at first. To visualize this throughput enhancement, one can look at the one source two sink network presented in Figure 7 of Reference [4]

and the corresponding discussion. A slightly different version of that network was later named as the butterfly network (see Figure 2 of Reference [5] for instance). With the help of the butterfly network, it was shown in [5] that the achievable network coding data rate is more than the routing data rate for multicast and multiple unicast flows. By extending the Max-flow Min-cut Theorem to network information flow, the work in [4] showed that optimum throughput of a multicast network can be achieved with network coding (which is not always possible with conventional routing). Reference [6] established that linear combination of incoming data packets at the intermediate nodes is sufficient to achieve multicast capacity, and this is termed linear network coding (LNC). A further illustration of this technique is random linear network coding (RLNC) [7] where the coding coefficients are chosen randomly from a finite field of sufficient size. Maximum achievable limit of capacity of a multicast network can also be attained with high probability with RLNC. Apart from the throughput and capacity benefit, network coding can improve latency, energy efficiency, network security, etc. in various networks and systems like lossy networks, ad-hoc sensor networks, delay tolerant networks, multicast networks, satellite communications, peer-to-peer content distribution systems, data storage systems, etc. [5,8–14].

A network is said to be solvable if each receiver in the network can recover their demands (set of packets, originated at the sources) from the received network coded packets [15]. When any link in a network carries a single data packet (raw or coded) and coding coefficients at each intermediate node are considered as vectors, it is termed as scalar network coding [16]. Similarly, when the links carry packet vectors instead of a single packet, and coding coefficients at intermediate nodes are considered as matrices, we call it vector network coding [16]. The latter has certain advantages over the previous linear network coding, such as higher ratio of total possible coding coefficients to total possible source messages [17]). Some of those advantages are summarized in [18]. The connection of field size with linear solvability of a network was first revealed in Reference [6], where it was shown that a multicast network is linearly solvable if the field size is sufficiently large. Since then, there has been extensive research on the effect of field size on scalar linear solvability and on the effect of field size and message dimension on vector linear solvability of various multicast and non-multicast network configurations [15,18–25]. Some recent work on this solvability issue has shown that not only the field size but also the order of the subgroups in the multiplicative group of the finite field are responsible for linear solvability of a multicast network [26]. For the rest part of this paper, by linear network code, we refer to scalar linear network code.

From the inception of RLNC, there are two basic modes of operation which are present in multicast or multiple unicast scenarios in a packet erasure wireless environment. For a large stream of data, it is quite a tough job to encode all packets of the stream because the computational complexity of encoding becomes pretty high. In a practical approach, data streams are divided into a fixed size of blocks and encoding is done within a block. These blocks are called generations [27], and the encoder moves to the next generation only when all packets of current generation are delivered to the receivers. This generation-based RLNC approach [28–36] has an inherent problem of large decoding delay because of the packet level coding. The second approach is slightly different from this and known as the systematic RLNC approach [37–40]. Here all packets of a generation are sent uncoded at first. After that, coded packets are sent to recover the packets which are lost due to channel erasure. The systematic approach is beneficial in a low loss network, but the performance becomes significantly poor with increase of erasure probability of channels, and the situation gets worse in terms of delivery delay when the receivers want in-order packet delivery. Several works are done in obtaining the decoding and delivery delay for both of the schemes from queueing as well as probabilistic point of view. Various works are also done to modify those approaches to overcome certain drawbacks or to make them fit to fulfill certain setup requirements or to pace up the transmission procedure [33,41,42].

While sending coded packets in the block RLNC or systematic RLNC approaches, the idea of *innovativeness* plays an important role. A coded packet is *innovative* to a receiver when the corresponding coefficient vector is not in the span of the coefficient vectors of previously received packets. When each receiver receives an *innovative* packet each time it receives a coded packet, we say the corresponding network coding strategy is throughput optimal. Those kinds of network coding strategies which produce *innovative* combinations in each attempt are said to have *innovation guarantee property* [43]. When we see a network where a coding strategy with *innovation guarantee property* is employed, data transmission in that network achieves optimal throughput and that network coding strategy is said to be a throughput optimal strategy. Li. et al. [6] showed that linear network codes are sufficient to achieve the optimal throughput in a multicast network. Now, if the network encoding strategy is not throughput optimal, the receivers need to collect some extra coded packets to fulfill their requirements even if the channels are erasure free. RLNC-based broadcast is an example of such a scenario where there is no issue of the network solvability. The broadcast shows near-optimal performance for a sufficiently large finite field [7]. A throughput optimal LNC scheme which deterministically chooses *innovative* coefficient for each packet in the linear combination by a polynomial time algorithm is known as deterministic LNC (DLNC) [44]. However, the DLNC algorithm is hard to implement without proper feedback from the receivers. One GF(2)-based throughput optimal LNC is proposed in [32] which can be equipped with block-based LNC. This is named triangular network coding, and it does not require any feedback from the receivers.

Effect of feedback can be exploited in LNC to increase the throughput optimality of encoding as well as to reduce the block level and packet level delay in various application-oriented scenarios. Some initial literature on this can be found in [45,46]. At the transmitter side, feedback from the receivers can be used to keep track of the receivers' progression in terms of the acquired degrees of freedom (dof) [47]. In DLNC, these dofs are used in the online algorithm to form *innovative* packets at each time slot. Following this idea, References [31] and [47] proposed two ARQ-based throughput optimal encoding techniques. These techniques were equipped with fixed-generation-based situations [31,48] as well as with stochastic arrival-based situations [36,47,49,50]. Some adaptive and opportunistic algorithms were given in [51,52] where the reduction of decoding delay was given more priority rather than throughput optimality, and algorithms of these type form a new class of network code, termed as instantly decodable network code (IDNC). References [50,53,54] analyzed similar feedback-based schemes respectively in deadline-aware applications, in delay-tolerant applications, and for dynamic transmission rate adaptation for improved throughput-delay performance.

1.1. Motivation

Queueing analysis is one of the key tools for analyzing various performance measures of a network coded multicast or broadcast system. Numerous literature is available where queueing analysis is done over different kinds of network coded systems to find out improvements in various issues like throughput and delay benefit; trade-offs between coding gain and decoding delay; characterization of capacity region, stability region and throughput region; limitation in feedback's usefulness etc. for various packet injection processes in sender queue (such as stochastic injection, constant data injection, and burst injection) [35–37,47,48,55–57]. Among those works, Reference [47] is the first to provide a technique that helps in keeping shorter expected sneder queue (SQ) lengths by exploiting the use of feedback.

In conventional generation-based multicast, the transmitter cannot drop any packet of a generation even if there is only one receiver left who has yet to gather some packets to decode the generation. This situation is exactly the same with throughput optimal and nonoptimal encodings. The transmitter drops the whole generation at a time when decoding has happened at each receiver. Therefore, this concept is termed the *drop when decoded* (DWD) scheme [43,47]. The main problem with the DWD technique is

the high probability of blocking new incoming packets to the SQ. Practically, the SQ is of finite length, and the transmitter cannot accommodate an incoming packet in SQ if it is full. No new arrival to SQ can be taken place to the fully occupied queue until the current generation under transmission is dropped. In systematic encoding approach, the situation is a bit better in terms of dropping a fraction of a generation, but that is not guaranteed always. This is because different multicast channels may behave differently in the same time slot, and this heterogeneity arises even when channels are stochastically homogeneous. Employment of feedback in the above methods helps the transmitter keep track of each receiver's degree of progression, but implementation of the scheme of dropping of packets from SQ before decoding at all receivers is not straightforward.

Sundararajan et al. came up with the notion of *seeing* a packet [43,47] and proposed the useful *drop when seen* (DWS) technique that, for the first time in the literature of network coding showed interest about dropping a packet before decoding the same at the receivers. They considered stochastic arrival at SQ and did a queueing theoretic analysis to show that, with their proposed scheme, the physical queue tracks the virtual queue [58,59] and the average queue length is minimized. The whole DWS technique is divided into two parts. One part does the queue management, and the other part does throughput optimal encoding. Theorem 8 and Theorem 11 of [43] establish the throughput optimality of the DWS technique and explain the conditions at which that optimality holds. The related discussion and results of [43] ensure that optimal throughput is obtained in a DWS-based single source, multiple sink multicasting, or broadcasting scenario.

Beside throughput optimality and efficient SQ management benefit, the issue of delivery and decoding delay is prominent in the DWS scheme. Analysis of the delay profile of this scheme was done extensively in [49,50]. Several works are done [50,53,60] to modify the DWS algorithm for delay control while keeping the main advantages of the technique. Reference [61] performed a statistical packet dropping analysis of a generation-based DWS broadcast system, and based on that analysis, provided a closer look at the delay profile for different decoding situations. A throughput-delay tradeoff is shown in [54] by careful and dynamic adaption of the transmission rate. Some other works adopted the idea of *seeing* in their proposed multicast models and showed the throughput and delay improvement of the models [34,35,55,57].

In DWS technique, the queue update module (QUM) at the sender performs efficient SQ management by keeping track of each receiver's *knowledge space* [43,47] through perfect feedback. The QUM does the whole operation at the end of each slot so that the coding module (CM) can perform throughput optimal encoding on the queue content at the beginning of the next time slot. However, to keep track of each receiver's *knowledge space*, QUM needs to do Gauss–Jordan elimination which is of high computational complexity. Thus the cost of efficient queue management of the DWS is paid in terms of high computational complexity. Moreover, the throughput optimal encoding algorithm which is executed at CM is just an instance of the polynomial time algorithm of DLNC encoding. Computational complexity of both the algorithms at QUM and CM is high and increases rapidly with the generation size, the number of the receivers and the size of the finite field under consideration in a generation-based DWS broadcast. This analysis is elaborately done in Section 3 of this paper. The above discussion motivates us to look for modified algorithms for QUM and CM which will be of less computational complexities, but the essence of the DWS technique will remain intact.

1.2. Contributions and Organization of the Paper

From the discussion of Section 1.1, one can infer that DWS algorithms are deterministic. In this paper, we propose two modified algorithms respectively for CM and QUM which are randomized. Hence, we name our modified technique as the *randomized drop when seen* (rDWS) technique. In a network coded multicast or broadcast, the receivers store the received packets at their buffer. When they gather sufficient

packets, decoding happens, and decoded packets are transferred to intended applications. In other words, decoding occurs when a receiver's *knowledge space* catches up with the *knowledge space* of the transmitter's. Now, each receiver keeps track of its own *knowledge space*. In our proposed rDWS scheme, we exploit this fact and the advantage of feedback simultaneously. The transmitter needs not keep track of the *knowledge space* of the receivers. Instead, the dimension information of the spaces is enough to perform efficient SQ management as in DWS case. Here, the transmitter gets the dimension information through feedback (we consider NACK-based feedback) from the receivers.

We design our QUM algorithm based on the above discussion. This modified algorithm is computationally less expensive than the one in DWS because, here, the transmitter needs not perform Gauss–Jordan elimination n times (n is the number of broadcasting receivers) to keep track of each receiver's *knowledge space*. Rather, the receivers compute their *knowledge space* at each slot and convey the dimension information of the spaces to the transmitter through feedback.

Now, for the coding module in rDWS case, throughput optimal encoding is not possible as the transmitter does not have the complete *knowledge space* information of the receivers. The QUM discards the packets which are *seen* by all receivers at the end of a slot. The CM does a random linear encoding with the next *unseen* packets (considering the next *unseen* packets of all receivers) at the beginning of the next slot. Therefore, our proposed algorithm is similar to conventional RLNC encoding [7]. Also, this randomized version of the CM algorithm is very simple and computationally less expensive than the one in the DWS case.

Employment of random linear encoding in our technique does not alter the network solvability issue [7]. However, we have to analyze the throughput efficiency of our randomized CM algorithm in terms of probability of getting an *innovative* combination, generated by the CM. We first analyze the probability of getting an *innovative* coefficient ($\mathcal{P}_C(i)$) for the i^{th} *unseen* packet of SQ. *Innovative* coefficient of an *unseen* packet $\mathrm{p_i}$ implies such a coefficient which is such that, when it is used in the linear combination by CM, dimension of each receiver's (whose next *unseen* packet is $\mathrm{p_i}$) *knowledge space* increases by one, upon successful reception of that combination. Mathematically, finding an exact expression of $\mathcal{P}_C(i)$ is found to be difficult. Therefore, we do two different analyses of $\mathcal{P}_C(i)$, namely, *maximum entropy* analysis and *average understanding* analysis, and provide a lower bound on $\mathcal{P}_C(i)$. Based on this, we perform an analysis of the probability of getting an *innovative* combination (or *innovative* coded packet), formed by CM.

As the idea of dropping a packet from SQ even before decoding at the receivers is also present in our proposed scheme, we carry out a statistical performance analysis on packet dropping as well on decoding by means of simulation. We consider three metrics, the cumulative packet dropping probability, the average time to drop the last packet of a generation, and the cumulative decoding probability of a generation in order to inspect the performance of the rDWS scheme. In [7], we saw that the conventional RLNC shows near-optimal performance for a finite field of sufficient size. Similar conclusions can be drawn for the rDWS technique from our analytical and simulation results.

The rest of the paper is organized as follows: Section 2 describes the system settings and provides some preliminary definitions. In Section 3, detailed description of the existing DWS technique, the proposed rDWS technique, and their comparison in terms of computational complexity are given. Section 4 is concerned with how close the rDWS encoding is towards throughput optimality. In Section 4.1, analysis on the probability of getting an *innovative* coefficient for an *unseen* packet in the linear combination is presented. Using this, the probability of a linear combination being *innovative* is analyzed in Section 4.2. The statistical performance evaluation of the rDWS technique through simulation is done in Section 5. Finally, we conclude the paper in Section 6.

2. System Model

We consider a broadcasting communication scenario similar to [43]. A sender is trying to broadcast a series of data packets to n receivers over independent, discrete time, packet erasure wireless channels. A receiver can detect an erasure in the transmission. Packets arrive at the sender from some information source according to some stochastic process and are stored in the finite-lengthed sender queue. These stored packets are divided into fix-sized generations for encoding purposes. We assume that the SQ has at least one generation of packets before broadcasting starts. Instead of sending raw packets, the sender does linear network encoding (over a finite field of sufficient size) within a generation and transmits the coded packets. Only a single packet (raw or coded) can be sent over a time slot. For ease of explanation, we consider generation-based encoding. However our proposal and its analysis are equally applicable to non-generation-based encoding.

Each receiver has a finite length buffer to store the incoming packets. When a receiver collects a sufficient number of appropriate coded packets, decoding happens, and decoded packets are delivered to the intended applications.

In this paper, we are dealing with two network coding techniques, DWS and rDWS. To execute these online techniques, the sender needs to know each receiver's state of progress at each time slot. Therefore, perfect error-free feedback is considered. In the DWS case, feedback is sent in the form of ACK from each receiver to convey successful reception (or no erasure in transmission). In the rDWS case, NACK is sent to indicate either erasure or reception of a *non-innovative* packet.

In Figure 1, we outline our model for the broadcasting transmission scenario we will be studying in this paper. As mentioned in Section 1.1, the modules QUM and CM process the data packets at the transmitter side, which are then sent to the receivers through wireless channels. The roles of these two modules will be discussed in detail in Section 3.

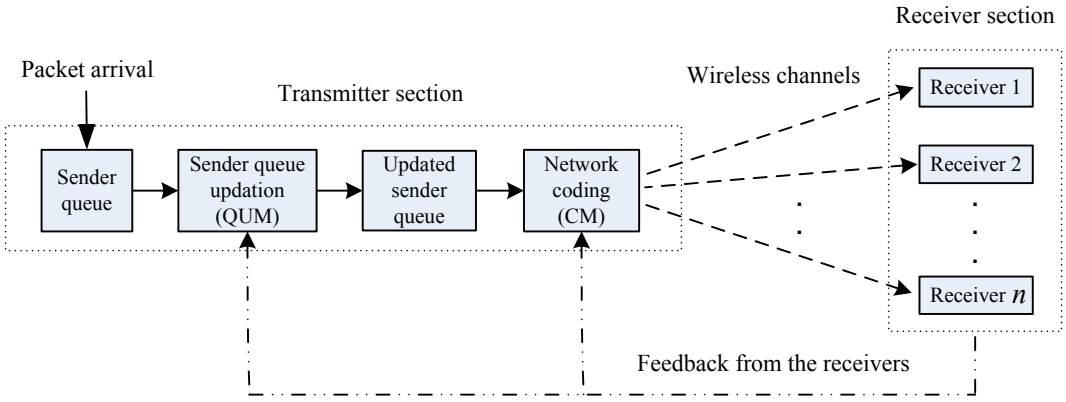


Figure 1. Block diagram of the components of the broadcasting transmission scenario.

Timing: We depict the timing occurrence of the events within a slot in Figure 2. The CM forms a linear combination using the next *unseen* packets of all the receivers at the beginning of a slot. The coded combination is sent just after that. We assume negligible propagation delay for simplicity. Also, the coded transmission reaches the receivers (if not erased at the channels) before they prepare and send feedback for that slot. Feedback from each receiver is received at the sender before the end of the same time slot. Hence, the feedbacks are guaranteed to convey current slot's reception information. At the end of the slot, SQ content is updated by the QUM according to feedback, and then the next *unseen* packets to the receivers are grouped for the next slot's encoding purpose.

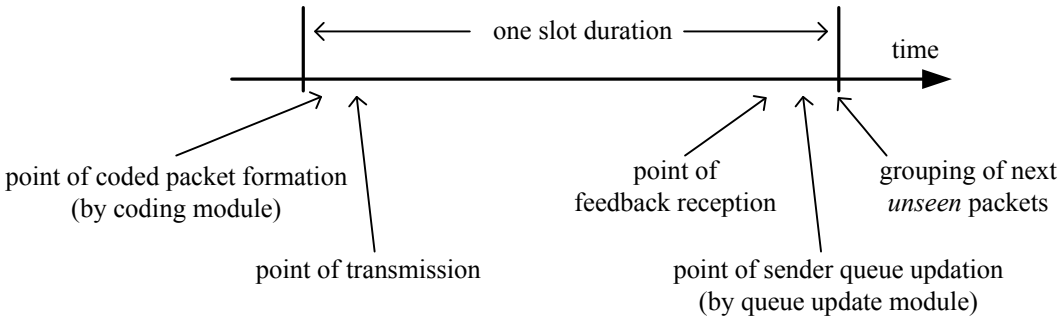


Figure 2. Relative timing of encoding, transmission, feedback reception, and sender queue updation points within a time slot.

Before going into the details of the techniques (DWS and rDWS), here we look at some preliminary definitions and necessary explanation.

Definitions

- *Packet index*: Following in-order packet delivery (i.e., the packets are delivered in the same order they arrive at the SQ), the i^{th} packet which arrives at the SQ is said to have index i.
- *Coefficient vector of a coded packet*: The vector corresponding to a network coded packet or coded combination which consists of the coefficient of the raw packets which are involved in that combination.
- *Knowledge space of a node*: The vector space at a node which is the span of the coefficient vectors of the available packets at that node.
- *Innovative packet*: A network coded packet is *innovative* to a receiver if the corresponding coefficient vector does not belong to the receiver's *knowledge space*. An *innovative* packet will always increase the dimension of a receiver's *knowledge space* upon successful reception.
- *Hearing a packet*: A receiver has *heard* of a packet $\mathrm{p_i}$ if it has received a linear combination involving $\mathrm{p_i}$.
- *Seeing a packet*: A receiver has *seen* a packet $\mathrm{p_i}$ if it can compute a linear combination of the form $\mathrm{p_i} + \sum_{\forall k} \alpha_k \mathrm{p_k}$ ($k > i$ and $\alpha_k \in \mathbb{F}_q$ for each k) based on the combinations it has received up to a slot.
- *Witness of a seen packet*: Linear combination of the form, $\mathrm{p_i} + \sum_{\forall k} \alpha_k \mathrm{p_k}$ ($k > i$ and $\alpha_k \in \mathbb{F}_q$ for each k), is the *witness* for receiver j of *seeing* $\mathrm{p_i}$ which is denoted by $w_j(i)$.

Example 1. *Suppose at a particular time slot SQ contains packets* $\mathrm{p_1}, \mathrm{p_2}, \mathrm{p_3}$ *and* $\mathrm{p_4}$ *and a receiver j has seen only* $\mathrm{p_1}$*. For j, the next unseen packet is* $\mathrm{p_2}$*. The CM makes a linear combination* $l = \alpha_1\mathrm{p_1} + \alpha_2\mathrm{p_2} + \alpha_3\mathrm{p_3} + \alpha_4\mathrm{p_4}$*, and broadcasts it along with the coefficient vector* $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ *in the header. On successful reception of l, j sends an ACK to the transmitter. Receiving the ACK, transmitter understands that j has seen* $\mathrm{p_2}$ *and has only heard of* $\mathrm{p_3}$ *and* $\mathrm{p_4}$*, and its next unseen packet is* $\mathrm{p_3}$*. The readers are encouraged to read [43] for more information.*

3. Existing DWS Technique and the Proposed rDWS Technique

As said before, our proposed technique is the randomized version of the conventional DWS scheme which is essentially a deterministic technique. Basic advantages of the DWS are two-fold. Firstly, it is throughput optimal, and secondly, it maintains a shorter expected SQ length throughout the transmission period. The essence of the DWS technique lies in the dropping of packets from SQ before decoding at receivers, and the QUM at the sender performs this operation. The other module, CM, does the throughput optimal encoding.

In the randomized version of the DWS method, we too have these two modules although their functions are modified. To understand the necessity of the rDWS technique and how we are incorporating randomization, we need to look at the DWS algorithms first. In Section 3.1, first we analyze the CM algorithm of the DWS technique. Then we provide the rDWS version of the algorithm in detail. A similar analysis of QUM algorithms (DWS and rDWS respectively) is given in Section 3.2.

The discussions of Sections 3.1 and 3.2 compare the DWS algorithms with the rDWS counterparts which establish the usefulness of our proposed scheme over DWS technique.

3.1. The Coding Module

3.1.1. DWS Case

In the DWS technique, the CM forms an *innovative* linear combination using the next *unseen* packets of the receivers. Algorithm 1 includes the work of CM in a step by step manner. Before the execution of the algorithm, CM groups the receivers with the same next *unseen* packets. Let $R(i)$ denote the set of receivers of which the next *unseen* packet is $\mathrm{p_i}$. In a generation-based encoding, each packet of the generation is the next *unseen* packet to at least one receiver in the worst-case situation. Therefore, we consider all possible $R(i)$ where i varies from 1 to m (m is the generation size). Algorithm 1 deterministically chooses coefficients for each *unseen* packet (coefficient of $\mathrm{p_i}$ in the combination is taken as γ_i) in such a way that it leads to an *innovative* combination, $l = \sum_{i=1}^{m} \mathrm{p_i}\,\gamma_i$. The CM finds a coefficient α_i ($\in \mathbb{F}_q$) to be *innovative* for $\mathrm{p_i}$ if it is different from the coefficient of $\mathrm{p_i}$ in y_j for each $j \in R(i)$. This ensures the increment of dimension of receiver j's *knowledge space* upon successful reception of the coded packet.

Algorithm 1 DWS Algorithm (The Coding Module)

1: Initialize $\gamma_i = 0$, for $i = 1$ to m
2: **for** $i = 1$ to m **do**
3: **for** $j = 1$ to $|R(i)|$ **do**
4: Initialize $y_j = 0$
5: **for** $k = 1$ to $i - 1$ **do**
6: $y_j \leftarrow y_j + \alpha_k\, w_j(k)$
7: **end for**
8: **end for**
9: **for** $j = 1$ to q **do**
10: **for** $k = 1$ to $|R(i)|$ **do**
11: **if** coefficient of $\mathrm{p_i}$ in $y_k = \alpha_j$ **then**
12: **Continue** loop for the next value of j at line 9
13: **end if**
14: **end for**
15: $\gamma_i = \alpha_j$
16: **Continue** loop for the next value of i at line 2
17: **end for**
18: **end for**
19: Compute the linear combination, $l = \sum_{i=1}^{m} \mathrm{p_i}\,\gamma_i$

Note that, if j is such a receiver that belongs to $R(i)$ but has not *heard* of packet $\mathrm{p_i}$, the corresponding y_j does not have a $\mathrm{p_i}$ term. Therefore any coefficient for $\mathrm{p_i}$ is going to be *innovative* for j. Thus, the fate of the coefficient of $\mathrm{p_i}$ in the final combination (l) depends on the receivers who have *heard* of $\mathrm{p_i}$ in any of the previous slots. We will discuss more on this while analyzing the rDWS technique in Section 4.

Now, we look for the worst-case computational complexity of Algorithm 1. Maximum value of $|R(i)|$ can be n which is the total number of receivers. At worst case, the loop starting at line 5 is executed $m-1$ times, and the algebraic expression of the witness of a specific packet in terms of that and other packets can contain at most m packets. Considering all these facts, overall complexity of Algorithm 1 is found to be $\mathcal{O}(mn(m^2+q))$. Therefore, from complexity perspective, one can see that finding an *innovative* combination is very expensive if the number of receivers and generation size are high or even moderate.

Minimum Field Size Requirement

The minimum field size requirement for the throughput optimal DWS encoding is the same as the number of receivers (Theorem 8 of Reference [43]).

3.1.2. rDWS Case

To reduce the computational complexity of Algorithm 1, we convert the previous deterministic algorithm into a randomized one. In conventional random linear network encoding [7] random coefficients are chosen from a finite field of sufficient size to form coded packets. This may not be optimal from the throughput efficiency perspective but is proven to be multicast capacity achieving. Here, the idea of RLNC is mixed with the original CM algorithm. The CM in the rDWS case picks random coefficients from a finite field for each next *unseen* packet of SQ and makes the linear combination (l). We present the CM algorithm of the rDWS technique in Algorithm 2 where the worst case scenario is considered as in the CM-DWS algorithm (which implies the involvement of all packets of a generation in the linear combination).

Algorithm 2 rDWS Algorithm (The Coding Module)

1: Initialize $\gamma_i = 0$, for $i = 1$ to m
2: **for** $i = 1$ to m **do**
3: Pick an α randomly from $\mathbb{F}_q$
4: $\gamma_i = \alpha$
5: **end for**
6: Compute the linear combination, $l = \sum_{i=1}^{m} \mathrm{p_i}\, \gamma_i$

We see the computational complexity of Algorithm 2 is $\mathcal{O}(m)$ which is independent of the total number of receivers or the field size. The reduction in computational complexity in CM algorithm from DWS to rDWS is obtained in the cost of employing *non-innovativeness* in encoding. Although it brings some extra delay in decoding, there is no issue from the network solvability aspect. Cruces et al. [30] formulated the probability of getting J *innovative* combinations out of K combinations ($K \geq J$) as a function of field size for random DWD encoding. Similarly, we analyze our rDWS encoding in Section 4 to track the effect of randomization in throughput efficiency.

Minimum Field Size Requirement

As rDWS encoding is not throughput optimal, there is no minimum field size requirement actually. However, one can notice that if the field size is less than the number of receivers, there may be some situations where the probability of getting an *innovative* combination is zero (i.e., no proper choice is available in the pool of choices, depending on the finite field, that can make an *innovative* combination). Keeping this in mind, for our analysis, we only consider the extension fields of GF(2) of which the sizes are greater than or equal to the number of receivers. As an example, in Figure 6 (Section 4.2.1), the maximum number of receivers under consideration is 15. The nearest extension field of GF(2) of which the size is

greater than or equal to 15 is GF(2^4). Hence, we plot the graphs in Figure 6 with respect to field size, starting from $2^4 = 16$.

3.1.3. Remarks

While describing the CM algorithms (Algorithms 1 and 2), we consider that all packets are involved in the linear combination l. This is not the case that always happens. At a particular slot, the loop starting at line 2 (Algorithms 1 and 2) is executed for those values of i which correspond to the index of the next *unseen* packets (for all receivers). In the worst case, i takes each value from 1 to m and the loop is executed m times. When all the receivers belong to a particular $R(i)$ for some $i \in \{1, 2, \cdots, m\}$, the CM sends $\mathrm{p_i}$ uncoded as there is no need to form a linear combination.

3.2. The Queue Update Module

3.2.1. DWS Case

As said before, the QUM drops the packets which are *seen* by every receiver from the SQ and keeps only those packets of which is still *unseen* to some receiver(s). However, to maintain the SQ content with only *unseen* packets, the sender needs to keep track of each receiver's *knowledge space*. A matrix of dimension $m \times m$ can hold one receiver's *knowledge space* information for a generation size m. Also, the CM of the DWS technique needs the witness information of each receiver to perform encoding operation (line 6, Algorithm 1), and the witness information can be obtained from the rref (reduced row echelon form) of the *knowledge space* matrices.

Although the job of QUM is to manage sender queue, some portion of the algorithm (line 7, Algorithm 3) is executed with the help of the receivers. If a receiver successfully receives a coded packet, ACK is generated in constant time (i.e., with $\mathcal{O}(1)$ complexity) and is sent back to the transmitter. For now, we focus our attention only on the computational complexity of the algorithm and keep aside the propagation delay through wireless channels. Therefore, ACKs are generated in a parallel manner at all receivers. Based on the feedback, QUM updates *knowledge space* matrices ($\mathbf{B}_1, \mathbf{B}_2, \cdots, \mathbf{B}_n$). By finding the common *knowledge space*, the sender gets information about the packets which are *seen* at all receivers. Next, it drops the packets from SQ and updates *knowledge space* matrices.

The QUM performs its operations at each time slot. Also, upon successful reception of a coded packet, a receiver *sees* its next *unseen* packet. These two facts tell us that set D (line 15, Algorithm 3) contains only one packet. Hence, only a single packet can be dropped from SQ at a single slot.

One can note that line 1 of the algorithm is executed only once (before the start of broadcasting). After that, it is run from line 2 which is being continued until the broadcast of the current generation is finished. When ACK is received from receiver j, the row of $\mathbf{B}_j$ corresponding to the next *unseen* packet of receiver j is updated with the coefficient vector of combination l, and there can be at most m components of that vector if all packets of the generation are involved in the creation of l. As the *knowledge space* matrices are of dimension $m \times m$, there can be at most m packets in the set D_j when all columns of its corresponding $\mathbf{B}_j$ matrix are pivot columns. Apart from these facts, the arithmetic complexity of Gauss–Jordan elimination (which is $\mathcal{O}(n^3)$ for a $n \times n$ matrix) and the complexity of Algorithm 1 lead us to the worst case complexity of Algorithm 3 which is found as $\mathcal{O}(mn(m^2 + q))$.

There are basically two reasons behind the high computational complexity of Algorithm 3. The first one is to keep track of the receivers' *knowledge space* and to perform Gauss–Jordan elimination. The second reason is the involvement of Algorithm 1 which is also computationally expensive. However, both of the facts are unavoidable as these are required to achieve the goals of the DWS technique. Now we present the QUM algorithm for rDWS scheme.

Algorithm 3 DWS Algorithm (The Queue Update Module)

1: Initialize $\mathbf{B}_1$, $\mathbf{B}_2$, $\cdots$, $\mathbf{B}_n$ as $m \times m$ zero matrices
2: **if** SQ is empty **then**
3: Do nothing
4: **else**
5: Call Coding Module (DWS), get linear combination l and transmit it
6: **end if**
7: ACK generation at receivers
8: **for** $j = 1$ to n **do**
9: **if** ACK is received from receiver j **then**
10: Update the row of $\mathbf{B}_j$ corresponding to the next *unseen* packet of receiver j with the coefficient vector corresponding to l
11: Perform Gauss–Jordan elimination of $\mathbf{B}_j$
12: **end if**
13: Find D_j which is the set of packets corresponding to the pivot columns of $\mathbf{B}_j$
14: **end for**
15: Find $D = \cap_{j=1}^{n} D_j$
16: **for** $j = 1$ to n **do**
17: $\mathbf{B}_j \leftarrow$ Change all elements of the pivot column and pivot row of $\mathbf{B}_j$ corresponding to the packet in D to zero
18: **end for**
19: Drop the packet in D from SQ

3.2.2. rDWS Case

In rDWS, to form a random linear combination, the CM does not require the *knowledge space* information of the receivers. However, to maintain the SQ with only *unseen* packets, the *knowledge space* information is needed. More precisely, information about dimension of the *knowledge spaces* is sufficient for the QUM to do the job. To decode a generation, each receiver tracks its own *knowledge space* and stores that information at each slot. Now, the dimension information of the spaces can be conveyed to the transmitter through feedback. Hence, instead of matrices, we take n variables as counters in the QUM algorithm to store the dimension information.

Likewise in Algorithm 3, a receiver can store its *knowledge space* in a matrix. Following the CM algorithm of the rDWS case, a receiver's *knowledge space* dimension will increase if the linear combination is *innovative* and the corresponding channel is not in erasure. A receiver sends a NACK to the sender if either it has received a *non-innovative* packet or the packet is erased in the channel. Here NACK is preferred instead of ACK (like in the DWS case) because, through NACK we can convey both the information, *non-innovativeness* and channel erasure. However, the NACK generation at receivers is not going to occur with constant complexity ($\mathcal{O}(1)$). Algorithm 4 describes the NACK generation procedure at receiver j. Line 1 of the algorithm is executed only once (before the start of broadcasting). After that, it is run from line 2 which is being continued until the broadcast of the current generation is finished. Line 6 and line 9 ensure that, even in rDWS case, all packets are *seen* (and delivered) in the same order they arrive at SQ. The computational complexity of NACK generation (Algorithm 4) is found as $\mathcal{O}(m^3)$ because of the Gauss–Jordan elimination step at line 5.

Algorithm 4 NACK generation at receiver j

Initialize $\mathbf{B}_j$ as a $m \times m$ zero matrix
if linear combination l is received at the receiver **then**
  Find D_1 which is the set of packets corresponding to the pivot columns of $\mathbf{B}_j$
  Update the row of $\mathbf{B}_j$ corresponding to the next *unseen* packet of receiver j with the coefficient vector corresponding to l
  Perform Gauss–Jordan elimination
  Find D_2 which is the set of packets corresponding to the *main diagonal* pivot columns (these are the pivot columns for which the pivot elements are in the main diagonal of a matrix) of updated $\mathbf{B}_j$
  if $D_2 = D_1$ **then**
    Send NACK
    Make all elements of the last updated row of $\mathbf{B}_j$ zero
  end if
else
  Send NACK
end if

From the feedback of the receivers, the transmitter gets the dimension information and updates the counters (k_j for $j = 1, 2, \cdots, n$). From the counters' value, the QUM identifies the packets which are *seen* by every receiver. Among these packets, the QUM drops the latest packet from SQ which was not dropped earlier. With the same argument given in the QUM-DWS case, one can observe that only a single packet can be dropped at a single slot from the SQ in rDWS case also.

The best part here is that the transmitter needs not keep track of the whole *knowledge space* of the receivers, but just the dimension. Also, NACK generation is a parallel process which can occur simultaneously at all receivers. This phenomenon reduces the overall computational complexity of the QUM algorithm of the rDWS which is presented in Algorithm 5.

Algorithm 5 rDWS Algorithm (The Queue Update Module)

Initialize $k_1, k_2, \cdots, k_n$ as counters with value zero
if SQ is empty **then**
  Do nothing
else
  Call Coding Module (rDWS), get linear combination l and transmit it
end if
NACK generation at receivers
for $j = 1$ to n **do**
  if NACK is received from receiver j **then**
    Do nothing
  else
    $k_j \leftarrow k_j + 1$
  end if
end for
Find k_{min} = minimum element of $\{k_1, k_2, \cdots, k_n\}$
Drop the packet with index k_{min} from SQ if it was not dropped at a previous slot

Next, we proceed for the computational complexity analysis of Algorithm 5. From line 2 to line 7, the computational complexity is $\mathcal{O}(m^3)$ (using the complexity of Algorithms 2 and 4). Complexity of line 16 is $\mathcal{O}(1)$. For the rest of the lines, the combined complexity is $\mathcal{O}(n)$ which is easy to deduce. Hence, the worst case complexity of Algorithm 5 is found as $\mathcal{O}(m^3 + n)$, which is less than the complexity of the QUM-DWS algorithm (Algorithm 3). This is a vast improvement.

While making this comparison between the DWS and rDWS techniques, the channel propagation delays of both the forward and feedback paths are kept aside. As the primary goal is to reduce the computational complexity while shifting from the DWS to rDWS method, some parameters like propagation delay are kept aside as they impart the same effect on the system in each case.

3.2.3. Remarks

A broadcast system which uses rDWS method to send coded packets differs from a system with the DWS technique in work distribution between the transmitter and the receivers for proper flow of data in the system. In the DWS case, the transmitter is dependent on the receivers just for ACK reception. Apart from this, the transmitter does all the jobs of packet preparation and data transmission. Receivers only receive those packets according to the channel condition and wait for the proper time to decode the packets. Whereas, in the rDWS case, the receivers also play a significant role in packet processing at the transmitter. Here, the transmitter does not have exact information about receivers' progress in terms of their *knowledge space*. The receivers keep themselves updated about their own *knowledge space* and convey the dimension information to the transmitter at the end of each slot. From the decoding point of view, this can be very useful if cooperation between receivers is allowed (though it is not always possible as the receivers may be situated very far from each other, geographically). Then they can help each other to speed up their progress in coded packet accumulation at very adverse channel (these are the channels through which they are connected with the transmitter) conditions.

An ACK-based QUM algorithm can also be constructed for the rDWS scheme. A receiver will send an ACK as feedback when it receives an *innovative* packet. One can infer that the computational complexity of an ACK-based QUM algorithm will be precisely the same as the computational complexity of Algorithm 5.

4. Throughput Efficiency Analysis of rDWS Technique

Because of the random selection of the coefficients, rDWS encoding is not throughput optimal. It was shown that, if we choose a random coefficient vector for a set of packets, the vector is going to be *innovative* with probability close to 1 with increasing field size [7]. In this section, our goal is to find out the probability with which a randomly picked-up coefficient vector is going to be *innovative* in rDWS scenario.

We pick up coefficients one by one for a linear combination in rDWS encoding. Each pick remains independent of others. Whether a coefficient is *innovative* depends on the receivers of which the next *unseen* packet is the one for which the coefficient is being chosen for (i.e., the *innovativeness* of the coefficient of $\mathrm{p_i}$ depends on $R(i)$).

4.1. Probability of Innovativeness of a Picked-up Coefficient

Here, we analyze the probability of getting an *innovative* coefficient for a packet which will be included in the linear combination l. This analysis will lead us to find the probability of getting an *innovative* linear combination, which is equivalent to examining the throughput efficiency of the rDWS encoding.

If one looks at Algorithm 1, from line 5 to line 7, the CM finds the expression of y_j for receiver $j \in R(i)$. There can be two types of receivers in $R(i)$. The receivers who have not *heard* of packet $\mathrm{p_i}$ are of the first type, and the set of these receivers is denoted as $T(i)$. The second type of receivers, who have *heard* of $\mathrm{p_i}$ but have not *seen* it, belongs to set $S(i)$. Therefore, $R(i) = T(i) \cup S(i)$. One can observe that $R(1) = T(1)$ as

$S(1)$ is always a null set. Apart from the receivers which belong to some $R(i)$ for $i = 1, 2, \cdots, m$, there may exist some receivers which have *seen* all packets of the generation. The set of those receivers is denoted as $E(m)$.

Now, the expression of y_j for a receiver $j \in T(i)$ does not contain a $\mathrm{p_i}$ related term. Therefore, any choice of coefficient for $\mathrm{p_i}$ in the linear combination is going to be *innovative* with respect to any receiver in $T(i)$. Next, for a receiver $j \in S(i)$, the corresponding y_j have a $\mathrm{p_i}$-related term and let the coefficient of $\mathrm{p_i}$ in y_j be $c_j(i)$. Let us also consider a vector $\mathbf{c(i)}$ that contains all the $c_j(i)$'s (for a fixed $S(i)$ and $j = 1, 2, \cdots, |S(i)|$). Therefore, $\mathbf{c(i)} = (c_1(i), c_2(i), \cdots, c_{|S(i)|}(i))$. All the $c_j(i)$'s in the vector $\mathbf{c(i)}$ may have the same values or may have different values from each other or any situation in between these two can occur. The total number of situations that can occur here is same as the integer partitions [62] of $|S(i)|$. When all coefficients are of the same value, total $q-1$ choices (excluding that value) will be a *good choice* for γ_i (line 4, Algorithm 2). By *good choice*, we mean that the corresponding chosen coefficient is *innovative*. In the extreme case, when all $c_j(i)$'s are different, the total $q - |S(i)|$ *good choices* are available. Therefore, the CM in rDWS case chooses an *innovative* coefficient with probability $\frac{1}{q-1}$ and $\frac{1}{q-|S(i)|}$ respectively in the terminal situations described above.

It is evident that, with rDWS technique, the probability of choosing an *innovative* coefficient for packet $\mathrm{p_i}$ (which we denote by $\mathcal{P}_C(i)$) only depends on $S(i)$ as that probability is always one with respect to any receiver in $T(i)$. For the same reason, $\mathcal{P}_C(1) = 1$ as $|S(1)|$ is always zero. Now we illustrate our discussion on integer partitions of $|S(i)|$ and the corresponding *good choices* available to the CM (rDWS) in the following example.

Example 2. *Let us consider a broadcasting scenario where the packets are encoded with the rDWS technique. At the beginning of a particular time slot, it is found that* $|S(3)| = 4$ *and the vector* $\mathbf{c(i)}$ *takes the form,* $\mathbf{c(3)} = (c_1(3), c_2(3), c_3(3), c_4(3))$. *Depending on the channel erasures and the time slot under consideration, total five forms of* $\mathbf{c(i)}$ *can happen. These are* $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, $(\alpha_1, \alpha_2, \alpha_3, \alpha_3)$, $(\alpha_1, \alpha_1, \alpha_2, \alpha_2)$, $(\alpha_1, \alpha_2, \alpha_2, \alpha_2)$ *and* $(\alpha_1, \alpha_1, \alpha_1, \alpha_1)$ *where* $\alpha_1, \alpha_2, \alpha_3, \alpha_4$ *are arbitrary and distinct elements of* $\mathbb{F}_q$. *In the case of* $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$, *all coefficients are different. Therefore, the total number of available good choices for* γ_3 *is* $q-4$ *and the probability of choosing an innovative coefficient for* $\mathrm{p_3}$ *is* $\dfrac{q-4}{q}$.

The form $(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$ *corresponds to the integer partition* $|S(3)| = 4 = 1+1+1+1$, *where the partition has a total of four different parts. Similarly the other forms of* $\mathbf{c(3)}$ *correspond to the partitions* $4 = 1+1+2$, $4 = 2+2$, $4 = 1+3$ *and* $4 = 4$ *respectively. For the partition* $4 = 2+2$, *two receivers of* $S(3)$ *have the same coefficient of* $\mathrm{p_3}$ *(say* α_1*) in their corresponding* y_j*'s. The rest of the two receivers also have the same coefficient (say* α_2*), but it is different from the previous one (i.e.,* $\alpha_1 \neq \alpha_2$*). Therefore, each part of a partition denotes how many of the receivers in* $S(3)$ *have the same coefficient of* $\mathrm{p_3}$ *in their corresponding* y_j *expressions. The total number of parts of a partition indicates the number of field elements which the CM must not pick as* γ_3 *in l. Thus, subtraction of the total number of parts of a partition from q gives total number of good choices for* γ_3. *Hence, the number of good choices available for both the partitions* $4 = 2+2$ *and* $4 = 1+3$ *is* $q-2$. *One can notice that, which receivers have the same coefficient of* $\mathrm{p_3}$ *does not affect the number of good choices, whereas how many of the receivers have the same coefficient matters. In Table 1, a detailed description of all the partitions of* $|S(3)|$ *is given.*

Table 1. The probability of *innovativeness* for all possible partitions of $|S(3)|$ when $|S(3)| = 4$.

Form of $\mathbf{c(3)}$	Partition of $\mathbf{|S(3)| = 4}$	No. of Parts	No. of *Good Choices* for γ_3	Probability of *Innovativeness*
$(\alpha_1, \alpha_2, \alpha_3, \alpha_4)$	$1+1+1+1$	4	$q-4$	$\frac{q-4}{q}$
$(\alpha_1, \alpha_2, \alpha_3, \alpha_3)$	$1+1+2$	3	$q-3$	$\frac{q-3}{q}$
$(\alpha_1, \alpha_1, \alpha_2, \alpha_2)$	$2+2$	2	$q-2$	$\frac{q-2}{q}$
$(\alpha_1, \alpha_2, \alpha_2, \alpha_2)$	$1+3$	2	$q-2$	$\frac{q-2}{q}$
$(\alpha_1, \alpha_1, \alpha_1, \alpha_1)$	4	1	$q-1$	$\frac{q-1}{q}$

4.1.1. Integer Partitions of $|S(i)|$ and $\mathcal{P}_C(i)$

So far, we have found out the probability of getting an *innovative* coefficient for $\mathrm{p_i}$ corresponding to a particular integer partition of $|S(i)|$. Let us construct a set $M(i)$ that contains all partitions of $|S(i)|$ where partitions are arranged in decreasing order of number of parts. Also, each partition is structured in decreasing order of parts. If two partitions are of same number of parts, the partition with the larger first part is placed after the other one. If the first part is same for two partitions, the partition with the larger second part is placed later. This procedure is continued until the tie regarding the position of the partitions is broken. In this way, for $|S(i)| = 7$, we get $M(i) = \{(1+1+1+1+1+1+1), (2+1+1+1+1+1), (2+2+1+1+1), (3+1+1+1+1), (2+2+2+1), (3+2+1+1), (4+1+1+1), (3+2+2), (3+3+1), (4+2+1), (5+1+1), (4+3), (5+2), (6+1), (7)\}$. The total number of elements of $M(i)$ is a function (ϕ) of $|S(i)|$ which we write as, $|M(i)| = \phi(|S(i)|)$. We consider another function $\psi(i,k)$ which gives number of parts of the k^{th} element of $M(i)$. One can see that $\psi(i,1) = |S(i)|$ and $\psi(i,|M(i)|) = 1$, for any i. Also, for a fixed i, the maximum value of $\psi(i,k)$ is $\psi(i,k)|_{\max} = \psi(i,1) = |S(i)|$, and minimum value of $\psi(i,k)$ is $\psi(i,k)|_{\min} = \psi(i,|M(i)|) = 1$.

We consider a random variable X_i which takes value k if the k^{th} element of $M(i)$ occurs. Let us define an event I which corresponds to getting an *innovative* coefficient for $\mathrm{p_i}$.

$$\begin{aligned}\mathcal{P}_C(i) = \Pr(I) &= \sum_{k=1}^{|M(i)|} \Pr(I \mid X_i = k) \Pr(X_i = k) \\ &\overset{(a)}{=} \sum_{k=1}^{|M(i)|} \left\{1 - \tfrac{\psi(i,k)}{q}\right\} \Pr(X_i = k) \\ &= \tfrac{1}{q} \sum_{k=1}^{|M(i)|} \{q - \psi(i,k)\} \Pr(X_i = k)\end{aligned} \tag{1}$$

where (a) follows from the inaugural discussion of Section 4.1. In Table 1, we have tabulated all the $\Pr(I \mid X_i = k)$'s for $|S(i)| = 4$. Next, we look for the probability of occurrence of a partition of $|S(i)|$ to get a complete expression of $\mathcal{P}_C(i)$ according to (1).

Probability of Occurrence of a Partition of $|S(i)|$

The probability $\Pr(X_i = k)$ depends on certain issues like broadcast settings, channel erasure probabilities, the time gap between the starting time slot of broadcasting and the slot under consideration. By broadcast settings, we mean the number of receivers, the size of the generation etc. These factors have an influence on $c_j(i)$ heterogeneity (the dissimilarity between the $c_j(i)$'s where $j = 1, 2, \cdots, |S(i)|$), which has direct impact on $\Pr(X_i = k)$.

If the erasure probabilities are very low and close to zero, the absolute $c_j(i)$ heterogeneity is very low. All the receivers are close to each other in terms of progress in getting coded packets, and the corresponding witness functions ($w_j(i)$ for $j = 1, 2, \cdots, |S(i)|$) are also similar. Therefore, we can say that a partition of $|S(i)|$ with more number of parts is less likely to occur than a partition with a fewer number of parts. In contrast, when erasure probabilities are high, $c_j(i)$ heterogeneity is also high. By similar logic, here a partition with more number of parts is more likely to occur than a partition with a fewer number of parts. These two are extreme situations that can happen. Any other situation may happen depending on the erasure probabilities and the situation lies in between the extreme cases. Hence, no perfect ordering between the $\Pr(X_i = k)$'s for $k = 1, 2, \cdots, |M(i)|$ exists.

If a considered time slot is beyond a particular value from the beginning of broadcasting, the $c_j(i)$ heterogeneity becomes prominent even for low channel erasure probabilities.

Now, for a suitable broadcast setting and moderate erasure probabilities, at a sufficiently large time slot, we focus our attention on the set $S(i)$. If we consider a partition of $|S(i)|$ as a state of the receivers in $S(i)$, the path through which the receivers have reached that state is a $|S(i)|$-dimensional random walk which is mathematically difficult to track [49,50]. Therefore, finding an exact expression of $\Pr(X_i = k)$ is clearly a daunting task. However, we perform three different analysis of $\Pr(X_i = k)$ to get some insight into the statistical understanding of $\mathcal{P}_C(i)$ according to (1).

4.1.2. Maximum Entropy of X_i and $\mathcal{P}_C(i)$

Each value the random variable X_i takes, corresponds to a partition of $|S(i)|$. From the previous discussion, it is clear that mathematically tracking the uncertainty of partitions is a tough job. Nevertheless, we can find the maximum uncertainty of the partitions. From the information theoretic view, maximum uncertainty of the partitions is same as the maximum entropy of the discrete random variable X_i (i.e., $H(X_i)$). It is well-known that $H(X_i)$ is maximum when X_i follows uniform distribution [63]. Thus, we get $\Pr(X_i = k) = \frac{1}{|M(i)|}$. Applying this in (1), we get the probability of choosing an *innovative* coefficient for packet $\mathrm{p_i}$ for maximum uncertainty of partitions as follows:

$$\mathcal{P}_{CM}(i) = \frac{1}{q|M(i)|} \sum_{k=1}^{|M(i)|} \{q - \psi(i,k)\} \tag{2}$$

In Figure 3, we have plotted the probability of *innovativeness* of a picked-up coefficient for packet $\mathrm{p_i}$ ($\mathcal{P}_C(i)$) with respect to field size for four different considerations of the probability $\Pr(X_i = k)$ (for a fixed $S(i)$ and $k = 1, 2, \cdots, |M(i)|$). The first consideration corresponds to the case where no perfect ordering of the $\Pr(X_i = k)$'s exists, and we generate the $\Pr(X_i = k)$'s for each k according to some arbitrary distribution in a *Monte Carlo* simulation environment. We plot the maximum and minimum values of $\mathcal{P}_C(i)$ among 10^7 runs according to (1) and denote these two values as *Random max.*, and *Random min.* in the figures. In the second case, we consider high $c_j(i)$ heterogeneity. This time, the $\Pr(X_i = k)$'s are also generated according to some arbitrary distribution with the consideration that a partition with more number of parts is more likely to occur than a partition with a fewer number of parts. To avoid ambiguity in choosing probability

of two partitions of same number of parts, we consider $\Pr(X_i = k) > \Pr(X_i = k+1)$ as the partitions are contained in $M(i)$ in such a way that, for any two consecutive partitions, the partition corresponding to more $c_j(i)$ heterogeneity is placed before the partition with less $c_j(i)$ heterogeneity. Again, we plot the maximum and minimum values of $\mathcal{P}_C(i)$ among 10^7 runs according to (1) and denote these two values as *Descending max.* and *Descending min.* in the figures. Similarly, *Ascending max.* and *Ascending min.* points in the plots denote the maximum and minimum values of $\mathcal{P}_C(i)$ for low $c_j(i)$ heterogeneity consideration. Lastly, $\mathcal{P}_{CM}(i)$ is plotted according to (2).

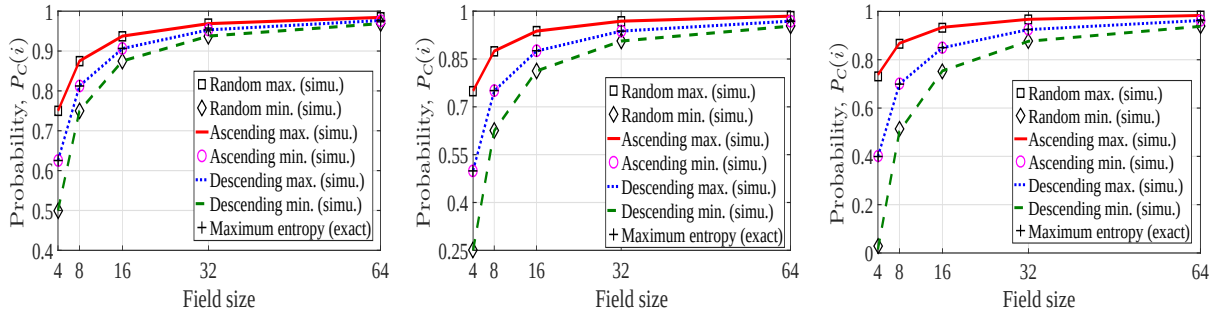


Figure 3. Probability of *innovativeness* of a picked-up coefficient, $\mathcal{P}_C(i)$, versus field size where $|S(i)| = 2$ (left subfigure), $|S(i)| = 3$ (middle subfigure) and $|S(i)| = 4$ (right subfigure).

There are three subfigures in Figure 3 where we have depicted the probabilities with respect to field size. From left to right, the subfigures correspond to $|S(i)| = 2$, $|S(i)| = 3$ and $|S(i)| = 4$ respectively. In each case, the probabilities increase as we increase the field size. Therefore, the chance of getting an *innovative* coefficient for $\mathrm{p_i}$ can be made arbitrarily close to one for a field of sufficient size. When we move from the left to the right subfigure, the $c_j(i)$ heterogeneity gets increased. As a result, the $\mathcal{P}_C(i)$'s decrease from left to right for a particular field size. For instance, $\mathcal{P}_C(i)$ values corresponding to *Descending max.* curve at field size 16 are 0.91, 0.88 and 0.85 (approximate values) respectively from the left to the right subfigure.

From Figure 3, we observe one more interesting fact. The overall range of the $\mathcal{P}_C(i)$ values for *Random case* is split by the *Maximum entropy* curve in each subfigure. The upper range corresponds to the $\mathcal{P}_C(i)$ values of *Ascending case*, whereas the lower range corresponds to the $\mathcal{P}_C(i)$ values of *Descending case*. This occurrence is because of the fact that maximum entropy corresponds to the uniform distribution which implies equal probability for each partition of $|S(i)|$. Thus, *Maximum entropy* case acts as the boundary between *Ascending* and *Descending* cases.

4.1.3. Average Understanding of *Innovativeness* and $\mathcal{P}_C(i)$

We have seen the likelihood of a partition depends on the erasure probabilities. Maximum entropy case implies maximum uncertainty of partitions, and this happens when all partitions are equally probable. If we consider the partition which is denoted by $X_i = |M(i)|$, only a single part is available. For a finite field of size q, the number of $\mathbf{c}(\mathbf{i})$ vectors that correspond to partition $X_i = |M(i)|$ is q. As a whole, total $q^{|S(i)|}$ vectors can result for a particular $|S(i)|$. To have an overall understanding of the receivers of $S(i)$, we should look at all these $q^{|S(i)|}$ vectors. As discussed previously, no perfect ordering between the $\Pr(X_i = k)$'s exists. Here, we assume a broadcast set up and a time instance for which all $q^{|S(i)|}$ vectors are equally probable. The analysis of the probability of *innovativeness* for such an assumption gives an average understanding of the system irrespective of the physical relevance of the assumption.

We take a new function $\xi(i,k)$ which gives the total number of $\mathbf{c}(\mathbf{i})$ vectors corresponding to the partition $X_i = k$. Using $\xi(i,k)$, we rewrite (1) to get the probability of *innovativeness* for uniform distribution of the $\mathbf{c}(\mathbf{i})$ vectors as:

$$\begin{aligned}\mathcal{P}_{CU}(i) &= \tfrac{1}{q}\sum_{k=1}^{|M(i)|}\{q-\psi(i,k)\}\,\xi(i,k)\;\tfrac{1}{q^{|S(i)|}}\\ &= \tfrac{1}{q^{|S(i)|+1}}\sum_{k=1}^{|M(i)|}\{q-\psi(i,k)\}\,\xi(i,k)\end{aligned} \tag{3}$$

Lemma 1. *Let us consider a partition of $|S(i)|$ as $\mathcal{Z} := \sum_{v=1}^{V} a_v b_v$, where part b_v is present a_v times and $b_v > b_{v+1}$ (for $\forall v \in \{1,2,\cdots,V-1\}$). If the finite field under consideration is $\mathbb{F}_q$, the total number of possible $\mathbf{c}(\mathbf{i})$ vectors corresponding to $\mathcal{Z}$ is given in (4), where $A = \sum_{v=1}^{V} a_v$, and ${}^qP_A$ represents A permutations of q.*

$$\begin{aligned}N_{\mathcal{Z}} = \tfrac{1}{a_1!\,a_2!\cdots a_V!}&\left\{\binom{|S(i)|}{b_1}\binom{|S(i)|-b_1}{b_1}\cdots\binom{|S(i)|-(a_1-1)b_1}{b_1}\right\}\cdot\left\{\binom{|S(i)|-a_1b_1}{b_2}\binom{|S(i)|-a_1b_1-b_2}{b_2}\cdots\right.\\ &\left.\binom{|S(i)|-a_1b_1-(a_2-1)b_2}{b_2}\right\}\cdots\left\{\binom{|S(i)|-a_1b_1-\cdots-a_{V-1}b_{V-1}}{b_V}\cdots\binom{|S(i)|-a_1b_1-\cdots-(a_V-1)b_V}{b_V}\right\}\cdot{}^qP_A\end{aligned} \tag{4}$$

Proof. For partition $\mathcal{Z}$, part b_v is present a_v times. Therefore for each v, there are a_v groups where each group consists of b_v receivers. The first group of b_1 receivers can be chosen from $|S(i)|$ receivers in $\binom{|S(i)|}{b_1}$ many ways. Considering the second group of b_1 receivers, they can be chosen in $\binom{|S(i)|-b_1}{b_1}$ many ways. Proceeding in this way, the last group of b_1 receivers can be chosen in $\binom{|S(i)|-(a_1-1)b_1}{b_1}$ many ways. All of these a_1 choices are independent because the broadcasting channels and the receivers are independent. Now, these a_1 choices can be permuted among themselves in $a_1!$ ways. Therefore, the total number of available choices for first a_1b_1 receivers is $\left\{\binom{|S(i)|}{b_1}\binom{|S(i)|-b_1}{b_1}\cdots\binom{|S(i)|-(a_1-1)b_1}{b_1}\right\}\frac{1}{a_1!}$. Similar arguments can be shown for the next $a_2b_2, a_3b_3, \ldots, a_Vb_V$ receivers.

Now, a group hall receivers withinave the same coefficient, $c_j(i)$, in their corresponding y_j equation. The total number of groups corresponding to partition $\mathcal{Z}$ is $A = \sum_{v=1}^{V} a_v$. Therefore, the total number of different coefficients corresponding to $\mathcal{Z}$ is also A. These coefficients are elements of finite field $\mathbb{F}_q$. In a pool of q field elements, A elements can be permuted in ${}^qP_A$ ways. Thus, we get the total number of possible $\mathbf{c}(\mathbf{i})$ vectors corresponding to $\mathcal{Z}$ as in (4). □

Example 3. *Let us consider a partition of $|S(i)| = 10$ as, $|S(i)| = 10 = \sum_{v=1}^{V=3} a_v b_v$, where $b_1 = 3, b_2 = 2, b_3 = 1$, and $a_1 = 1, a_2 = 2, a_3 = 3$. If the size of the finite field under consideration is $q = 11$, from (4), we get the total number of $\mathbf{c}(\mathbf{i})$ vectors corresponding to the partition given above as, 4,191,264,000.*

Lemma 1 provides the total number of possible $\mathbf{c}(\mathbf{i})$ vectors corresponding to partition $\mathcal{Z}$. As $\mathcal{Z}$ is a partition of $|S(i)|$, $\mathcal{Z}$ corresponds to a partition $X_i = k$ for some $k \in \{1,2,\cdots,|M(i)|\}$. Therefore, we can find out $\xi(i,k)$ for each k using Lemma 1. Next, from (3), we get the probability $\mathcal{P}_{CU}(i)$.

4.1.4. Lower Bound on $\mathcal{P}_C(i)$

In this section, we provide a lower bound on the probability $\mathcal{P}_C(i)$. From (1) we get,

$$\begin{aligned}
\mathcal{P}_C(i) &\overset{(a)}{\geq} \frac{1}{q}\sum_{k=1}^{|M(i)|}\left[q-\max_{\forall k}\{\psi(i,k)\}\right]\Pr(X_i=k)\\
&\overset{(b)}{=} \frac{1}{q}\sum_{k=1}^{|M(i)|}\{q-\psi(i,1)\}\Pr(X_i=k)\\
&\overset{(c)}{=} \frac{1}{q}\sum_{k=1}^{|M(i)|}\{q-|S(i)|\}\Pr(X_i=k)\\
&\overset{(d)}{=} \frac{\{q-|S(i)|\}}{q}\sum_{k=1}^{|M(i)|}\Pr(X_i=k)\\
&\overset{(e)}{=} \frac{q-|S(i)|}{q}
\end{aligned} \tag{5}$$

where (a) is straightforward, (b) and (c) follow from the discussion of Section 4.1.1, (d) is true because the taken out term is constant over k, and (e) is correct because one of the partitions must occur. Equation (5) gives a lower bound on $\mathcal{P}_C(i)$ which we write as:

$$\mathcal{P}_{CL}(i) = \frac{q-|S(i)|}{q} \tag{6}$$

Intuitively, one can understand that the probability of *innovativeness* is minimum when the partition $X_i = 1$ results. At this situation, the number of *good choices* left is minimum. The lower bound is important because it helps to find the maximum time required to *see* a packet for a receiver. As decoding means *seeing* all packets of a generation [61], we can find maximum time to decode a generation for a receiver as well as for the whole receiver system. In short, $\mathcal{P}_{CL}(i)$ helps in characterizing the decoding delay [49,50,53] profile of the broadcasting system.

In Figure 4, we have plotted the probability of *innovativeness* of a picked-up coefficient for packet $\mathrm{p_i}$ with respect to field size for the maximum entropy case ($\mathcal{P}_{CM}(i)$), the average understanding case ($\mathcal{P}_{CU}(i)$), and the lower bound ($\mathcal{P}_{CL}(i)$). There are three subfigures where we have depicted the probabilities with respect to field size. From left to right, the subfigures correspond to $|S(i)| = 2$, $|S(i)| = 3$ and $|S(i)| = 4$ respectively. In each case, the probabilities increase as we increase the field size. Therefore, the chance of getting an *innovative* coefficient for $\mathrm{p_i}$ can be made arbitrarily close to one for a field of sufficiently large size. When we move from the left to the right subfigure, the $c_j(i)$ heterogeneity increases. As a result, the $\mathcal{P}_C(i)$ value decreases from left to right for fixed field size. For instance, the $\mathcal{P}_{CU}(i)$ values corresponding to average understanding curve at field size 16 are 0.88, 0.82, 0.77 (approximate values) respectively from the left to the right subfigure.

One can observe that, the $\mathcal{P}_{CM}(i)$ value is greater than the $\mathcal{P}_{CU}(i)$ value in each subfigure for fixed field size. Therefore, the maximum uncertainty of the partitions of $|S(i)|$ leads to better probabilistic performance than the average case scenario in terms of *innovativeness* of the chosen coefficient.

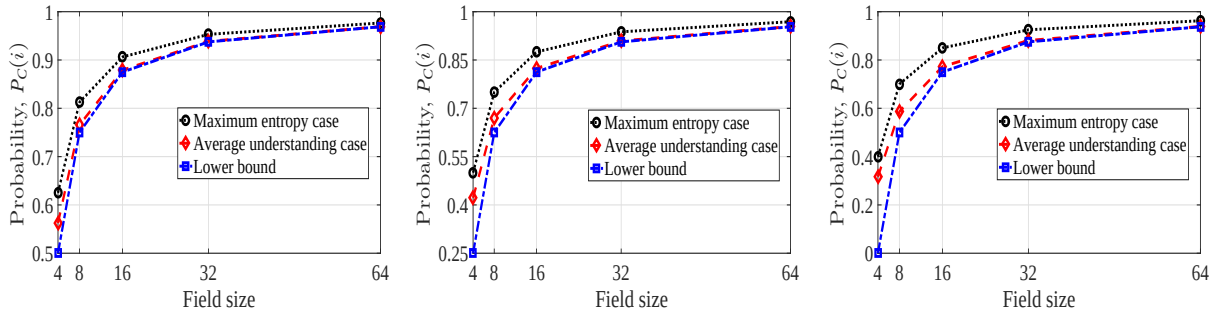


Figure 4. Probability of *innovativeness* of a picked-up coefficient ($\mathcal{P}_{CM}(i)$, $\mathcal{P}_{CU}(i)$ and $\mathcal{P}_{CL}(i)$) versus field size where $|S(i)| = 2$ (left subfigure), $|S(i)| = 3$ (middle subfigure) and $|S(i)| = 4$ (right subfigure).

4.2. Probability of Innovativeness of a Linear Combination

So far we have analyzed the probability of *innovativeness* of a coefficient corresponding to a packet which is a part of the final linear combination. Based on that analysis, here our goal is to find out the probability of *innovativeness* of the linear combination to get an insight on the throughput performance of the rDWS encoding.

Let us consider the CM is making a combination l with only two *unseen* packets, $\mathrm{p}_{\mathrm{i}_1}$ and $\mathrm{p}_{\mathrm{i}_2}$, at a particular time slot. The probability of *innovativeness* of the combination l ($\mathcal{P}_L$) is same as the joint probability of *innovativeness* of the coefficients of $\mathrm{p}_{\mathrm{i}_1}$ and $\mathrm{p}_{\mathrm{i}_2}$ (the coefficients are γ_{i_1} and γ_{i_2} respectively) which we denote as $\mathcal{P}_C(i_1, i_2)$. Now, the *innovativeness* probability of γ_{i_1} depends on the receivers of $R(i_1)$, whereas the *innovativeness* probability of γ_{i_2} depends on the receivers of $R(i_2)$. For any two receivers j_1 and j_2 (where $j_1 \in R(i_1)$ and $j_2 \in R(i_2)$), the channels through which they are connected to the transmitter are independent. Also, no co-operation is allowed between j_1 and j_2. Therefore we infer that, $\mathcal{P}_C(i_1, i_2) = \mathcal{P}_C(i_1) \cdot \mathcal{P}_C(i_2)$.

In the worst-case situation, the final combination l contains all packets of the generation. Though the worst-case situation may not be very likely to happen, to compare rDWS with DWS from a generation-based broadcasting perspective, it is considered that l consists of all packets. Now, extending the previous argument for two packets ($\mathrm{p}_{\mathrm{i}_1}, \mathrm{p}_{\mathrm{i}_2}$) to all m packets of the generation, we obtain the probability of *innovativeness* of a linear combination as follows:

$$\mathcal{P}_L = \mathcal{P}_C(1, 2, \cdots, m) = \prod_{i=1}^{m} \mathcal{P}_C(i) \overset{(a)}{=} \frac{1}{q^{m-1}} \prod_{i=2}^{m} \sum_{k=1}^{|M(i)|} \{q - \psi(i,k)\} \Pr(X_i = k) \tag{7}$$

where (a) follows from (1) and $\mathcal{P}_C(1) = 1$. DWS is a throughput optimal technique where each linear combination made by CM is *innovative*. The $\mathcal{P}_L$ helps in analyzing how close the rDWS encoding is towards the optimal DWS encoding.

4.2.1. Special Cases

In Section 4.1, we carried out three individual analyses of $\mathcal{P}_C(i)$. If we put the expression of $\mathcal{P}_{CM}(i)$ in place of $\mathcal{P}_C(i)$ in (7), we get the probability of *innovativeness* of l corresponding to Section 4.1.2, and we denote the probability as $\mathcal{P}_{LM}$. Similarly, we obtain the expression of $\mathcal{P}_L$ with respect to Sections 4.1.3 and 4.1.4 which we respectively denote as $\mathcal{P}_{LU}$ and $\mathcal{P}_{LL}$. Among these probabilities, here, we emphasize

$\mathcal{P}_{LL}$ as this particular probability provides a deeper understanding of the least efficient throughput performance of the rDWS encoding. From (6) and (7), we get the expression of $\mathcal{P}_{LL}$ as follows:

$$\mathcal{P}_{LL} = \frac{1}{q^{m-1}} \prod_{i=2}^{m} \{q - |S(i)|\} \tag{8}$$

Analysis of $\mathcal{P}_{LL}$ and the Minimum Value

So, it is evident that $\mathcal{P}_{LL}$ provides a lower bound on the probability of getting an *innovative* linear combination. It can be noted from (8) that $\mathcal{P}_{LL}$ depends on the field size and on the $|S(i)|$'s for $i = 2, 3, \cdots, m$. If one can find the minimum value of $\mathcal{P}_{LL}$, it will lead us to the worst-case behavior of the rDWS technique in terms of throughput efficiency. Therefore, in this section, we aim to find out the minimum value of $\mathcal{P}_{LL}$. The probability of *innovativeness*, $\mathcal{P}_C(i)$, is always one with respect to a receiver who is *unheard* of its next *unseen* packet p_i. While searching for the minimum value of $\mathcal{P}_{LL}$, we consider that each $|S(i)|$ is a nonzero integer (except for $|S(1)|$ which is always zero). This is to make sure that the probability of choosing an *innovative* coefficient for none of the packets of the generation is one, and each $\mathcal{P}_C(i)$ (except for $\mathcal{P}_C(1)$ which is always one) has some contribution in $\mathcal{P}_{LL}$. Also, none of the $|S(i)|$'s ($i = 2, 3, \cdots, m$) can be more than $n - m + 2$ as the number of receivers is n.

For fixed values of number of receivers n, generation size m, and field size q, finding the minimum value of $\mathcal{P}_{LL}$ is a standard optimization problem of the following form:

$$\begin{aligned} &\text{Minimize:} && \tfrac{1}{q^{m-1}} \prod_{i=2}^{m} \{q - |S(i)|\} \\ &\text{Subject to:} && \sum_{i=2}^{m} |S(i)| \leq n \end{aligned} \tag{9}$$

It is a nonlinear integer programming which we reframe as the following:

$$\min \quad f(\mathbf{x}) = \prod_{i=1}^{\mathcal{M}} (q - x_i) \qquad \text{s.t.} \quad \sum_{i=1}^{\mathcal{M}} x_i \leq \mathcal{N},$$

where,

$$\mathcal{M} = m - 1, \mathcal{N} = n, \mathbf{x} = (x_1, x_2, \cdots, x_{\mathcal{M}}), x_i = |S(i+1)|, x_i \in \{1, 2, \cdots, \mathcal{N} - \mathcal{M} + 1\} \text{ for } i \in \{1, 2, \cdots, \mathcal{M}\} \tag{10}$$

To solve the integer program, first we look into the following consideration. The objective function $f(\mathbf{x})$ has the same value for two vectors $\mathbf{x_1}$ and $\mathbf{x_2}$ if they are just element-wise permutation of one another. We call $\mathbf{x_1}$ and $\mathbf{x_2}$ as *equivalent* vectors. For instance, $(3, 1, 2)$, $(3, 2, 1)$, $(1, 3, 2)$, $(2, 3, 1)$, $(2, 1, 3)$ and $(1, 2, 3)$ form a group of *equivalent* vectors in the three-dimensional space. In a group of *equivalent* vectors, the vector whose elements are ordered in a non-increasing order from the first to the last element is denoted as the *representative* of that group. Thus, $(3, 2, 1)$ is the *representative* of the previously mentioned *equivalent* vectors.

The nonlinear programming problem is solved with a method which is quite similar to the famous *Branch and Bound* method [64]. A tree is constructed with only the *representatives* of each possible *equivalent* vector group which is depicted in Figure 5 for $\mathcal{M}$-dimensional case. Starting node of the tree is $\mathbf{x} = (1, 1, \cdots, 1)$ because $x_i \in \{1, 2, \cdots, \mathcal{N} - \mathcal{M} + 1\}$ for each i and the lower bound on the constraint is $\sum_{i=1}^{\mathcal{M}} x_i = \mathcal{M}$. Second level of the tree corresponds to constraint $\sum_{i=1}^{\mathcal{M}} x_i = \mathcal{M} + 1$, and only one node corresponding to vector $(2, 1, 1, \cdots, 1)$ is possible. Therefore, the node $(2, 1, \cdots, 1)$ is the only

child of $(1,1,\cdots,1)$. A node, $\mathbf{x} = (x_1, x_2, \cdots, x_{\mathcal{M}})$, at level h (which corresponds to the constraint $\sum_{i=1}^{\mathcal{M}} x_i = \mathcal{M} + h - 1$) can have at most $\mathcal{M}$ children at level $h+1$ which are $(x_1+1, x_2, \cdots, x_{\mathcal{M}})$, $(x_1, x_2+1, \cdots, x_{\mathcal{M}}), \cdots, (x_1, x_2, \cdots, x_{\mathcal{M}}+1)$. But, the number of children is much less because we only keep the *representative* vectors.

Now, we look at the positioning of the children of a particular parent node. The child whose first component of its vector is the largest among all is going to be the leftmost child. If the first component is same for two or more children, the child with the second largest component will be the leftmost child. This procedure continues until the leftmost child is decided. Once the leftmost position is decided, we proceed for the second leftmost position and the same rule is followed. We iteratively apply the rule to get the position of all children of a parent at each level and eventually obtain the complete tree structure as in Figure 5.

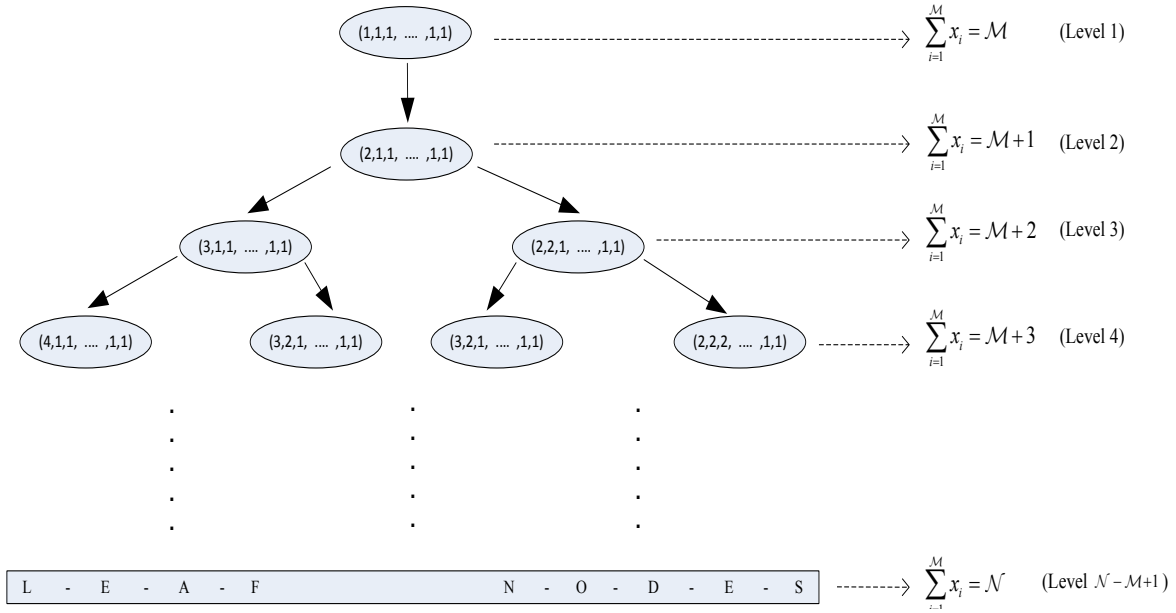


Figure 5. Tree corresponding to Section 4.2.1 where *representative* vectors represent nodes. The constraint corresponding to each level of the tree is written at the right.

The leaf nodes of the tree correspond to the constraint $\sum_{i=1}^{\mathcal{M}} x_i = \mathcal{N}$. Now, our goal is to find the node or the nodes for which the cost of the objective function $f(\mathbf{x})$ is minimum.

Lemma 2. *Cost of $f(\mathbf{x})$ associated with a node at level h is greater than the cost of $f(\mathbf{x})$ corresponding to any of its children.*

Proof. Let the cost of $f(\mathbf{x})$ of a node at level h is $\mathcal{W}_h$. We arbitrarily choose one of its children (which are at level $h+1$), and consider the associated cost as $\mathcal{W}_{h+1}$. The relation between $\mathcal{W}_h$ and $\mathcal{W}_{h+1}$ is,

$$\mathcal{W}_{h+1} = \frac{q-x-1}{q-x}\,\mathcal{W}_h \tag{11}$$

Depending on the position of the parent node at level h and the position of the considered child at level $h+1$, the value of x is from the set $\{1, 2, \cdots, \mathcal{N}-\mathcal{M}\}$. As the field size is greater than or equal to the number of receivers (Minimum field size requirement, Section 3.1.2), we get $q \geq \mathcal{N}$. Also, $\mathcal{M} \geq 2$ because generation size must be greater than or equal to two. Now from (11), we get $\mathcal{W}_{h+1} < \mathcal{W}_h$. □

Lemma 3. *The node at level h corresponding to vector* $(h, 1, \cdots, 1)$ *is the leftmost node at that level, and it has a child which is associated with vector* $(h+1, 1, \cdots, 1)$ *at level* $h+1$.

Proof. The proof is trivial, and we omit it. □

Proposition 1. *Starting from level 1 of the tree in Figure 5, we obtain the following order:*

$$f(\mathbf{x})|_{\mathbf{x}=(1,1,\cdots,1)} > f(\mathbf{x})|_{\mathbf{x}=(2,1,\cdots,1)} > f(\mathbf{x})|_{\mathbf{x}=(3,1,\cdots,1)} > \cdots > f(\mathbf{x})|_{\mathbf{x}=(\mathcal{N}-\mathcal{M}+1,1,\cdots,1)} \tag{12}$$

Proof. This directly follows from Lemma 2 and 3. □

Proposition 2. *The leftmost node at level h corresponds to the minimum cost of the objective function* $f(\mathbf{x})$ *at that level.*

Proof. We prove this by induction on level h.

Base case: For level 1 and 2, single nodes are present. So, the proposition is true by default for level 1 and 2. In level 3, there are two nodes which are respectively associated with vectors $(3, 1, 1, \cdots, 1)$ and $(2, 2, 1, \cdots, 1)$. Cost function associated with the first vector is $f(\mathbf{x})|_{\mathbf{x}=(3,1,\cdots,1)} = (q-3)(q-1)(q-1)^{\mathcal{M}-2} = (q^2-4q+3)(q-1)^{\mathcal{M}-2}$ whereas, the same for the second vector is $f(\mathbf{x})|_{\mathbf{x}=(2,2,\cdots,1)} = (q-2)(q-2)(q-1)^{\mathcal{M}-2} = (q^2-4q+4)(q-1)^{\mathcal{M}-2}$. Clearly, $f(\mathbf{x})|_{\mathbf{x}=(3,1,1,\cdots,1)} < f(\mathbf{x})|_{\mathbf{x}=(2,2,1,\cdots,1)}$. From Lemma 3, vector $(3, 1, 1, \cdots, 1)$ is associated with the leftmost node at level 3. So, the base case is true.

Induction hypothesis: Let, at level $h = k\,(k > 3)$, cost of $f(\mathbf{x})$ for vector $(k, 1, \cdots, 1)$ (which is the leftmost node at level k according to Lemma 3) is $\mathcal{W}'_k$. Now, we choose any arbitrary node at level k which is different from the leftmost node and consider its associated cost as $\mathcal{W}''_k$. Let us assume, $\mathcal{W}'_k < \mathcal{W}''_k$.

Inductive step: If we choose the cost associated with the leftmost node at level $k+1$ as $\mathcal{W}'_{k+1}$, then from Lemma 3 and the tree construction rule we get,

$$\begin{aligned} \mathcal{W}'_{k+1} &= \tfrac{q-x'-1}{q-x'}\,\mathcal{W}'_k \\ \text{or,} \quad \mathcal{W}'_k &= \tfrac{q-k}{q-k-1}\,\mathcal{W}'_{k+1} \end{aligned} \tag{13}$$

Now, we have to choose an arbitrary node at level $k+1$ which is different from the leftmost node. Without any loss of generality, here we choose a node which is a child of the arbitrarily chosen node (with cost $\mathcal{W}''_k$) at the induction hypothesis section. Let, the cost of the selected child node at level $k+1$ is $\mathcal{W}''_{k+1}$. The relation between $\mathcal{W}''_k$ and $\mathcal{W}''_{k+1}$ is,

$$\mathcal{W}''_k = \frac{q-x''}{q-x''-1}\,\mathcal{W}''_{k+1} \tag{14}$$

where, $x'' < k$ according to the tree construction rule. From the induction hypothesis,

$$\begin{aligned} & \mathcal{W}'_k < \mathcal{W}''_k \\ \text{or,} \quad & \tfrac{q-k}{q-k-1}\,\mathcal{W}'_{k+1} = \tfrac{q-x''}{q-x''-1}\,\mathcal{W}''_{k+1} \\ \text{or,} \quad & \tfrac{\mathcal{W}'_{k+1}}{\mathcal{W}''_{k+1}} = \tfrac{q^2-(k+x''+1)q+kx''+x''}{q^2-(k+x''+1)q+kx''+k} < 1 \\ \text{or,} \quad & \mathcal{W}'_{k+1} < \mathcal{W}''_{k+1} \end{aligned} \tag{15}$$

□

Theorem 1. *The node with the minimum cost of the objective function,* $f(\mathbf{x}) = \prod_{i=1}^{\mathcal{M}}(q - x_i)$ *with respect to the constraint,* $\sum_{i=1}^{\mathcal{M}} x_i \leq \mathcal{N}$ *corresponds to the vector* $\tilde{\mathbf{x}} = (\mathcal{N} - \mathcal{M} + 1, 1, 1, \cdots, 1)$*, where* q *is the field size,* $m = \mathcal{M} + 1$ *= generation size,* $n = \mathcal{N}$ *= number of receivers,* $\mathbf{x} = (x_1, x_2, \cdots, x_{\mathcal{M}})$*, and* $x_i \in \{1, 2, \cdots, \mathcal{N} - \mathcal{M} + 1\}$ *for* $i \in \{1, 2, \cdots, \mathcal{M}\}$.

Proof. We can prove the theorem using Lemma 3, Propositions 1 and 2. □

Corollary 1. *The set of optimal points of the nonlinear integer program in (10) is given as:*

$$\mathcal{Q} = \{\mathbf{x}^* \mid \mathbf{x}^* \text{ is a vector obtained by permuting the elements of the vector } \tilde{\mathbf{x}} = (\mathcal{N} - \mathcal{M} + 1, 1, \cdots, 1)\} \tag{16}$$

Proof. This follows from Theorem 1 and the idea of *equivalent* vectors. □

With the help of Corollary 1 and (9), we get the minimum value of $\mathcal{P}_{LL}$ as:

$$\mathcal{P}_{LL}|_{\min} = \frac{(q-1)^{m-2}}{q^{m-1}}(q - n + m - 2) \tag{17}$$

While solving the nonlinear program in (9), we have considered there is no such receiver which has not received any packet or which has received all the packets of the generation. This is to ensure that we obtain the lowest possible value of $\mathcal{P}_{LL}$ which is given in (17). If there are $|R(1)|$ receivers who have not received any packet and $|E(m)|$ receivers who have received every packet, the objective function of the integer program will remain same as in (9), but the constraint will be changed to, $\sum_{i=2}^{m} |S(i)| \leq n - |R(1)| - |E(m)|$. Here, we will also be able to find the minimum value of the objective function in the same manner described above, but this value will be greater than the value obtained in (17). So, we conclude that (17) provides the absolute minimum value of $\mathcal{P}_{LL}$.

We plot the minimum value of the lower bound $(\mathcal{P}_{LL})$ according to (17) in Figure 6 with respect to field size. In the left subfigure, the total number of receivers (n) is fixed at 15 while we vary the generation size (m) from 3 to 15 with an interval of 3. Similarly, in the right subfigure, m is fixed at 3, and n is changed from 3 to 15 with interval 3. The left graph shows that for a fixed group of receivers the lower bound gets improved (though minimal improvement) with the increment of generation size for a fixed field. So, larger generation size can be beneficial from the $\mathcal{P}_{LL}$ perspective. But, a large generation size will increase the encoding complexity. Hence, generation size optimization is required according to the application where rDWS scheme is incorporated. In contrast, the right graph shows that the lower bound gets deteriorated (significant deterioration) with the increase of the number of receivers. This happens because the $c_j(i)$ heterogeneity increases with the number of receivers.

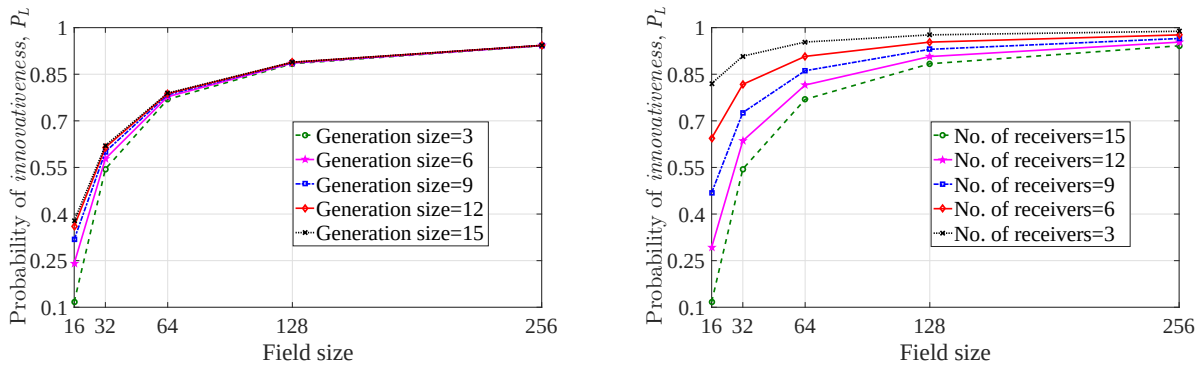


Figure 6. Minimum value of the lower bound of the probability of *innovativeness* ($\mathcal{P}_{LL}|_{\min}$) of a picked-up linear combination versus field size. For the left subfigure plots, we have considered $n = 15$ and $m = 3$, $m = 6$, $m = 9$, $m = 12$, $m = 15$, and for the right subfigure plots, we have considered $m = 3$ and $n = 3$, $n = 6$, $n = 9$, $n = 12$ and $n = 15$.

Secondly, the minimum value of the lower bound $(\mathcal{P}_{LL})$ is close to one for each plot of both graphs at field size 256. So, the chance of getting an *innovative* combination for rDWS encoding approaches certainty with the increase of field size. In general, this is true for any generation size and any number of receivers. So we conclude that, apart from the computational complexity benefit of our proposed rDWS technique, its encoding achieves near-optimal performance for a finite field of sufficient size. This fact also ascertains that the rDWS encoding is identical to the conventional random linear network encoding [7].

A Practical Consideration and the Minimum Value of $\mathcal{P}_{LL}$

In Section 4.1, we have seen that $R(i) = T(i) \cup S(i)$ (except for $i = 1$ where $R(1) = T(1)$). There it was argued that the probability of *innovativeness* is always one with respect to any receiver of $T(i)$. But, if the CM chooses the zero element of the field under consideration as γ_i, the receivers of $T(i)$ will fail in *seeing* $\mathrm{p_i}$ upon successful reception of the coded combination. So practically, the number of available *good choices* with respect to any receiver of $T(i)$ is $q - 1$.

For $i \neq 1$, let us look at $\mathcal{P}_C(i)$ for the following considerations:

Case 1. *When* $|R(i)| = |S(i)|$ *(i.e.,* $|T(i)| = 0$*), we get* $\mathcal{P}_C(i) \geq \frac{q-|R(i)|}{q}$ *(from (5)).*

Case 2. *Next, let* $|T(i)| = 1$ *and* $|S(i)| = |R(i)| - 1$*. In the worst situation, all receivers in* $S(i)$ *have different* $c_j(i)$*'s in their corresponding* y_j *equation and none of them are zero. The CM must not choose any of these* $c_j(i)$*'s (because of the receivers in* $S(i)$*) or the zero element of* $\mathbb{F}_q$ *(because of the receiver in* $T(i)$*) as* γ_i*. Hence, the total number of available good choices with respect to the receivers in* $R(i)$ *is* $(q - (|R(i)| - 1) - 1) = q - |R(i)|$*. Subsequently, we get* $\mathcal{P}_C(i) \geq \frac{q-|R(i)|}{q}$*.*

Case 3. *Next, we consider* $|T(i)| = 2$ *and* $|S(i)| = |R(i)| - 2$*. With similar argument given in case 2, we get* $\mathcal{P}_C(i) \geq \frac{q-|R(i)|+1}{q}$*.*

If we proceed in this way, as the terminal case we obtain $|R(i)| = |T(i)|$ (i.e., $|S(i)| = 0$). Here, it is evident that $\mathcal{P}_C(i) = \frac{q-1}{q}$.

Summarizing the above discussion it can be concluded that, for $i \neq 1$, the lower bound on the probability of *innovativeness* of a coefficient is $\mathcal{P}'_{CL}(i) = \frac{q-|R(i)|}{q}$, whereas for $i = 1$, the probability of *innovativeness* is $\mathcal{P}'_{C}(1) = \frac{q-1}{q}$. Using these findings, we obtain a lower bound on the probability of *innovativeness* of the linear combination as:

$$\mathcal{P}'_{LL} = \frac{q-1}{q^m} \prod_{i=2}^{m} \{q - |R(i)|\} \tag{18}$$

Our next goal is to search for the minimum value of $\mathcal{P}'_{LL}$. Likewise in Section 4.2.1, here the minimum value can be obtained by solving the following nonlinear integer programming:

$$\begin{aligned} &\text{Minimize:} && \tfrac{q-1}{q^m} \prod_{i=2}^{m} \{q - |R(i)|\} \\ &\text{Subject to:} && \sum_{i=2}^{m} |R(i)| = n - |E(m)| - |T(1)| \end{aligned} \tag{19}$$

Following the method of solution presented in Section 4.2.1, we get the minimum value of the lower bound as $\frac{(q-1)^{m-1}(q-n+m+|E(m)|+|T(1)|-2)}{q^m}$. One can infer that the absolute minimum value of $\mathcal{P}'_{LL}$ is obtained when $|E(m)| = 0$ and $|T(1)| = 1$ (as $|T(1)|$ can not be zero) which we write as:

$$\mathcal{P}'_{LL}\big|_{\min} = \frac{(q-1)^{m-1}}{q^m}(q-n+m-1) \tag{20}$$

In Figure 7, we plot the minimum value of the lower bound $\mathcal{P}'_{LL}$ according to (20) in the same manner as we did for $\mathcal{P}_{LL}|_{\min}$ in Section 4.2.1. The results of Figures 6 and 7 are very similar as there is not much difference in the values of $\mathcal{P}_{LL}|_{\min}$ and $\mathcal{P}'_{LL}|_{\min}$. So, similar conclusions can be drawn from Figure 7 as we have discussed for Figure 6 at the end of Section 4.2.1.

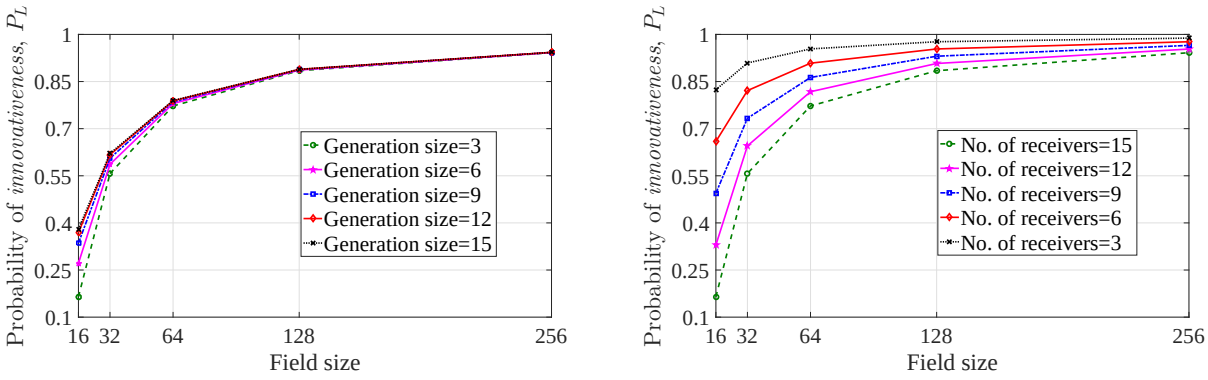


Figure 7. Minimum value of the lower bound of the probability of *innovativeness* ($\mathcal{P}'_{LL}\big|_{\min}$) of a picked-up linear combination versus field size. For the left subfigure plots, we have considered $n = 15$ and $m = 3$, $m = 6$, $m = 9$, $m = 12$, $m = 15$, and for the right subfigure plots, we have considered $m = 3$ and $n = 3$, $n = 6$, $n = 9$, $n = 12$ and $n = 15$.

5. Performance Evaluation of rDWS Scheme in Terms of Dropping and Decoding Statistics

So far, we have discussed the rDWS algorithms and analyzed the throughput efficiency of the scheme to compare it to the state of the art. As mentioned previously, while developing the rDWS technique it is ensured that, the sender does not necessarily lose its packet dropping capability even before decoding at receivers like DWS technique. The modified QUM algorithm ensures that. In this section, we evaluate

the statistical performance of packet dropping and packet decoding of our proposed scheme by means of simulation.

We create a *Monte Carlo* simulation environment where the model described in Section 2 is implemented in a discrete time, single transmitter, multiple receiver broadcasting scenario with packet erasure channels. The channels are independent and offer homogeneous, Bernoulli packet loss. ARQ is realized from the receivers to the transmitter through perfect, error-free feedback channels. As said before, packets are delivered (which also implies packets are *seen*) to the receivers in the same order they arrive at the SQ. So, packet p_i is *seen* before p_{i+1}. While doing the simulations, we assume negligible propagation delay and negligible processing time for various operations (like encoding at the sender, ACK/NACK generation at the receivers etc.) compared to a time slot.

The metrics which we consider for the statistical performance evaluation of the rDWS technique are the cumulative packet dropping probability, the average time to drop the last packet of a generation at the sender, and the cumulative packet decoding probability at the receivers. These metrics for rDWS depend on the size of the field, which is used to form the coded combinations. We consider the extension fields of GF(2) and perform the simulations with increasing field size. Finally, the DWS technique is simulated with the same setup, and the results are compared with the rDWS counterparts to get a comprehensive picture of the performance of the proposed approach. For the DWS consideration, the corresponding finite field is also chosen to be an extension field of GF(2) with the constraint that the field size is greater than or equal to the number of receivers. While doing the throughput efficiency analysis of the rDWS technique, we have also restricted our attention only to the extension fields of GF(2) whose size is greater than or equal to the number of receivers. But, here we relax that constraint for two basic reasons. First one is to observe how rDWS-based broadcast performs with small finite fields. Second reason is to show the usefulness of the rDWS over DWS as the later technique must not be performed with a finite field of size lesser than the number of receivers.

We conduct 10^5 independent runs for each evaluation scenario. The results are averaged over all runs and are plotted in the respective figures from Figures 8–10.

5.1. Packet Dropping Statistics

The QUM drops a packet from SQ when all receivers have *seen* it. At an arbitrary time slot t, let the state of a receiver is the index of its last *seen* packet. Clearly, the QUM is in a position to drop the packet with index same as the minimum state of all receivers at t.

5.1.1. Cumulative Dropping Probability

First, we look at the dropping probability of the packets of a generation. For 15 broadcasting receivers and erasure probability 0.5, we plot the cumulative dropping probability (with respect to time) of the packets of a generation with generation size 3 in Figure 8. Here, the cumulative probability of dropping a packet at t implies the total probability of dropping that packet from slot 1 to slot t. As there is no actual field size requirement for rDWS technique (Section 3.1.2), we perform the simulations for the respective fields: GF(2^2), GF(2^3), GF(2^4), GF(2^5) and GF(2^6). Finally, the cumulative dropping probability is plotted for DWS case where the field, GF(2^4) is considered (as total number of receivers is 15).

Figure 8 consists of three subfigures. From left to right, the subfigures correspond to the cumulative probability for the first, second and third packet of the generation respectively. For each subfigure, at a particular time slot, the dropping probability for rDWS cases increases and gradually becomes closer to the same for DWS case with the increase of field size. The CM of the rDWS technique picks random coefficients for the linear combination, and the chance of getting *innovative* coefficients increases with increasing field size. As a consequence, the chance of state transition for a receiver increases which in turn

enhances the packet dropping probability. From these observations, it is clear that the dropping probability for rDWS case is going to be very close to the same of the DWS case for a field of sufficiently large size, and that size directly depends on the broadcast settings. Here in our plots, one can observe that the dropping probability with respect to GF(2^6) for rDWS case is in fact very close to the DWS case at each time slot (in every subfigure).

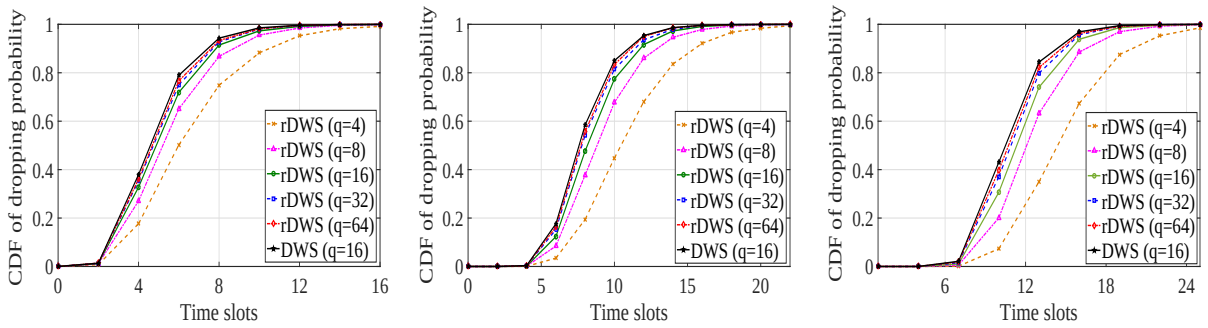


Figure 8. Cumulative probability of packet dropping for the first packet (left subfigure), the second packet (middle subfigure), and the third packet (right subfigure) versus time slots with $n = 15$, $m = 3$ and $p_e = 0.5$.

In Figure 8, the dropping probability curve for a particular rDWS case (or for the DWS case) grows at a relatively slower rate towards unity as we move from the left to the right subfigure. This is the result of the in-order delivery assumption. Packets are *seen* by the receivers in the same order they arrive at SQ, and the cumulative probability grows at a slower pace from the first to the third packet.

5.1.2. Average Dropping Time for the Last Packet of a Generation

Next, we evaluate the average number of time slots required to drop the last packet of a generation and plot these values with respect to erasure probability in Figure 9. Because of the in-order packet delivery, dropping the last packet implies dropping the whole generation. So, the average dropping time for the last packet is essentially identical to the average time to evacuate the generation from SQ.

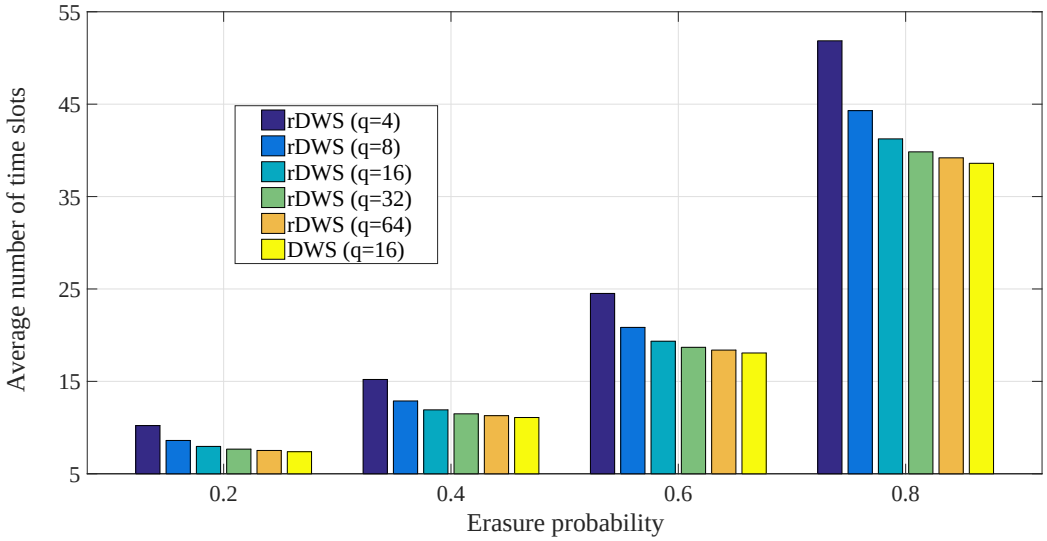


Figure 9. Average time to drop the last packet of a generation versus channel erasure probability with $n = 15$ and $m = 4$.

The simulations are performed for 15 broadcasting receivers and generation size four. We plot the average number of time slots for four erasure probabilities, 0.2, 0.4, 0.6 and 0.8. In each erasure probability case, rDWS simulations are done for fields, GF(2^2), GF(2^3), GF(2^4), GF(2^5) and GF(2^6), and DWS simulation is done for GF(2^4). Though the erasure probability is increased with an interval of 0.2 starting from 0.2, the average number of slots to drop the generation does not increase in an uniform manner. As an example, for rDWS with field GF(2^2), the average number of slots are 10, 15, 25 and 52 (approximate values) respectively. This is happening due to the rapid increase of erasure heterogeneity at a time slot with erasure probability. Though the channels are homogeneous in nature from erasure probability perspective, at a particular slot, whether a channel is in erasure or not is independent from the other channels. This phenomenon brings the slot-wise erasure heterogeneity for the channels which is increased in a non-uniform and rapid manner with the increase of erasure probability.

For each erasure probability consideration, the average number of slots for rDWS tends towards the DWS case as we move from GF(2^2) to GF(2^6). One can argue similarly as in Section 5.1.1 and conclude that DWS refers to the limiting behavior of the rDWS cases with increasing field size. In our evaluation, the performance of the rDWS scheme with GF(2^6) is sufficiently close to the DWS scenario. One more point to notice from Figure 9 is: the plots (all five plots for rDWS and the DWS plot) get sparse from each other as we increase the erasure probability. This is also due to the non-uniform and rapid increase of slot-wise erasure heterogeneity with erasure probability.

Now, the sender is able to drop the last packet of the generation indicates that all receivers have *seen* all packets of the generation, and the generation can be decoded. Thus, the generation size divided by the average time to drop the last packet of the generation provides the throughput of the whole broadcasting system. For instance, throughput of the rDWS broadcasting system (from Figure 9) with GF(2^2) for erasure probability 0.8 is $4/52 = 0.077$ packets/ slot (approximate value). So, Figure 9 provides an indirect measure of the throughput of the systems under consideration. Higher the average number of slots, lesser the throughput is.

5.2. Generation Decoding Statistics

While broadcasting the packets of a generation, the *knowledge space* of the transmitter is considered to be of dimension m. When a receiver *sees* a packet, the dimension of its *knowledge space* increases by one. The receiver can decode the generation when the dimension of its *knowledge space* becomes equal to the same of the transmitter's i.e., the *knowledge space* matrix (Section 3.2) of that receiver attains full-rank.

In this section, we investigate the cumulative probability of decoding a generation in two different contexts. In the first context, we find the decoding probability for an arbitrarily chosen receiver. Secondly, we find the same for the whole receiver system. Obviously, decoding of a generation by the receiver system is equivalent to decoding by the receiver (or receivers) who is the last receiver to *see* packet p_m.

We plot the cumulative decoding probability of a generation with respect to time in Figure 10, for a generation of size four. Total 15 broadcasting receivers are considered as before, and the erasure probability is 0.5. In the left subfigure, decoding probability for an arbitrary single receiver is depicted, whereas in the right subfigure, decoding probability for the whole receiver system is plotted. For both subfigures, the simulations are performed for five instances of the rDWS case and for the DWS case (same as in Section 5.1), and the results are plotted in each case.

For a fixed rDWS case (or for the DWS case), the cumulative probability curve for single receiver grows faster than the curve for all receiver in Figure 10. This is evident as decoding by all receivers is equivalent to decoding by the last receiver in the list. In both of the subfigures, the performance of the rDWS scheme with GF(2^6) is very close to the performance of the DWS technique.

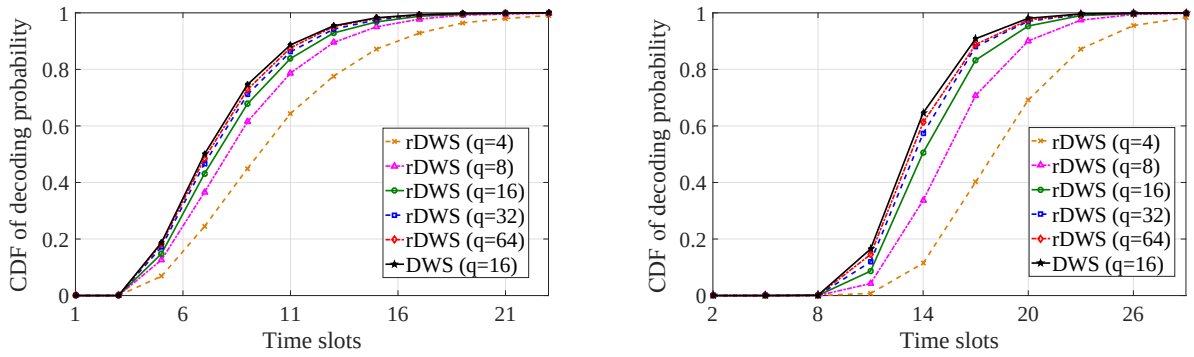


Figure 10. Cumulative probability of decoding a generation for an arbitrarily chosen receiver (left subfigure), and for the whole receiver system (right subfigure) with $n = 15$, $m = 4$ and $p_e = 0.5$.

6. Conclusions

In this paper, we have proposed a randomized version of the existing DWS technique for network coded broadcast. The main goal of this modified technique is to reduce the computational complexity of the existing algorithms while keeping the essence of the original technique intact. So, the notion of *seeing* a packet and the transmitter's ability to drop a packet before decoding at receivers are also present in our rDWS scheme. We exploit the fact that the receivers always keep track of their own *knowledge space*. Instead of keeping each receiver's complete *knowledge space* information, the transmitter gathers the dimension of the spaces through feedback (in the form of NACK). Implementation of this logic leads to much simpler and computationally less expensive algorithms. But, the price of this achievement is paid in the form of non-optimal encoding. From throughput efficiency analysis of the rDWS technique, we obtain a lower bound on the *innovativeness* probability of the coded combination with the consideration that all packets of the generation are used in the linear combination. Our theoretical and numerical performance analyses imply that rDWS achieves throughput optimality in an asymptotic sense with the increase of field size.

When we have a strong processor with large enough memory, DWS offers required performance though it involves high computational complexity. However, as future generation wireless technology might require lightweight, fast, and simple algorithms for packet processing, rDWS might just turn out to be a better choice. One can observe that the possible increment in number of retransmissions and consequent increase in latency with rDWS technique can easily be compensated with larger field size (Figures 8–10). As it is demonstrated in Figure 9, for an erasure probability 0.2, even with a smaller field of size 8, rDWS exhibits only 16.51% larger average latency than the same of the DWS with field size 16. If we allow an equal field size, the difference gets reduced to 7.78%, and with a larger field of size 32 for rDWS, it gets reduced to just 3.81%. For an erasure probability 0.8, with a smaller field of size 8, rDWS exhibits only 14.76% larger average latency than the same of the DWS with field size 16. If we allow an equal field size, the difference gets reduced to 6.85%, and with a larger field of size 32 for rDWS, it gets reduced to just 3.21%. So, with realistic and moderate to even bad channel conditions, one can have an almost optimal performance with proposed lightweight, low complexity rDWS algorithms. A large field and generation size is relevant to desktop-based applications, whereas embedded platforms and high-end smartphone-based applications require low to moderate field and generation size [65]. Most of the security-related operations require high generation and field size, but video streaming and P2P file sharing require low to moderate values of field and generation size [65]. It is assumed that a low to moderate field and generation size will lead to low complexity algorithms which are compatible with next-generation wireless technology vision and requirements. Here, rDWS can be the candidate of choice

as the same, while remaining a low complexity technique and using field sizes same to that of the DWS, provides near-optimal (almost as good as DWS) performance at moderate to difficult channel conditions.

As an immediate future extension of this work, a thorough mathematical analysis of the packet dropping and decoding statistics of the rDWS scheme can be done. A similar analysis for DWS technique was performed in [61]. Though we have carried out the same statistical analysis in Section 5 through simulation, a mathematical analysis of the same will help in getting a strong theoretical underpinning about the performance of the rDWS scheme.

Generation-based network coding is less suitable for real-time broadcast. So, another challenge is to search for a modified version of the rDWS scheme which will be ideal for real-time broadcasting system.

As the third extension of our work, a queueing theoretic analysis can be done with a stochastic arrival to analyze the issues related to the SQ stability.

Author Contributions: Conceptualization, S.G. and R.R.; Methodology, S.G. and R.R.; Software, S.G.; Validation, S.G. and R.R.; Formal analysis, S.G.; Investigation, S.G.; Resources, S.G. and R.R.; Data curation, S.G. and R.R.; Writing–original draft preparation, S.G.; Writing–review and editing, S.G. and R.R.; Visualization, S.G. and R.R.; Supervision, R.R.; Project administration, R.R.; Funding acquisition, Not Applicable.

Funding: This research received no external funding.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

LNC	Linear network coding
RLNC	Random linear network coding
DLNC	Deterministic linear network coding
IDNC	Instantly decodable network coding
SQ	Sender queue
DWD	Drop when decoded
DWS	Drop when seen
rDWS	Randomized drop when seen
QUM	Queue update module
CM	Coding module

References

1. Luby, M. LT codes. In Proceedings of the 43rd Annual IEEE Symposium on Foundations of Computer Science, Providence, RI, USA, 21–23 October 2002, pp. 271–280. [CrossRef]
2. Shokrollahi, A. Raptor codes. *IEEE Trans. Inf. Theory* **2006**, *52*, 2551–2567. [CrossRef]
3. Pakzad, P.; Fragouli, C.; Shokrollahi, A. Coding schemes for line networks. In Proceedings of the International Symposium on Information Theory, Adelaide, Australia, 4–9 September 2005; pp. 1853–1857. [CrossRef]
4. Ahlswede, R.; Ning Cai.; Li, S.R.; Yeung, R.W. Network information flow. *IEEE Trans. Inf. Theory* **2000**, *46*, 1204–1216. [CrossRef]
5. Fragouli, C.; Le Boudec, J.Y.; Widmer, J. Network coding: An instant primer. *ACM SIGCOMM Comput. Commun. Rev.* **2006**, *36*, 63–68. [CrossRef]
6. Li, S.R.; Yeung, R.W.; Ning, C. Linear network coding. *IEEE Trans. Inf. Theory* **2003**, *49*, 371–381. [CrossRef]
7. Ho, T.; Medard, M.; Koetter, R.; Karger, D.R.; Effros, M.; Shi, J.; Leong, B. A Random Linear Network Coding Approach to Multicast. *IEEE Trans. Inf. Theory* **2006**, *52*, 4413–4430. [CrossRef]

8. Ho, T.; Leong, B.; Medard, M.; Koetter, R.; Chang, Y.-H.; Effros, M. On the utility of network coding in dynamic environments. In Proceedings of the International Workshop on Wireless Ad-Hoc Networks, Oulu, Finland, 31 May–1 June 2004; pp. 196–200. [CrossRef]
9. Widmer, J.; Le Boudec, J.Y. Network coding for efficient communication in extreme networks. In Proceedings of the 2005 ACM SIGCOMM workshop on Delay-tolerant networking, Philadelphia, PA, USA, 26 August 2005; pp. 284–291.
10. Lucani, D.E.; Stojanovic, M.; Medard, M. Random Linear Network Coding For Time Division Duplexing: When To Stop Talking And Start Listening. In Proceedings of the IEEE INFOCOM, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 1800–1808. [CrossRef]
11. Joshi, G.; Liu, Y.; Soljanin, E. On the Delay-Storage Trade-Off in Content Download from Coded Distributed Storage Systems. *IEEE J. Sel. Areas Commun.* **2014**, *32*, 989–997. [CrossRef]
12. Sipos, M.; Gahm, J.; Venkat, N.; Oran, D. Erasure Coded Storage on a Changing Network: The Untold Story. In Proceedings of the 2016 IEEE Global Communications Conference (GLOBECOM), Washington, DC, USA, 4–8 December 2016; pp. 1–6. [CrossRef]
13. Gkantsidis, C.; Rodriguez, P.R. Network coding for large scale content distribution. In Proceedings of the IEEE 24th Annual Joint Conference of the IEEE Computer and Communications Societies, Miami, FL, USA, 13–17 March 2005; Volume 4, pp. 2235–2245. [CrossRef]
14. Cai, N.; Yeung, R.W. Secure network coding. In Proceedings of the IEEE International Symposium on Information Theory, Lausanne, Switzerland, 30 June–5 July 2002; p. 323. [CrossRef]
15. Ambadi, N. Multicast Networks Solvable over Every Finite Field. In Proceedings of the IEEE International Symposium on Information Theory (ISIT), Vail, CO, USA, 17–22 June 2018; pp. 236–240. [CrossRef]
16. Ebrahimi, J.; Fragouli, C. Vector network coding algorithms. In Proceedings of the IEEE International Symposium on Information Theory, Austin, TX, USA, 12–18 June 2010; pp. 2408–2412. [CrossRef]
17. Das, N.; Rai, B.K. Linear Network Coding: Effects of Varying the Message Dimension on the Set of Characteristics. *arXiv* **2019**, arXiv:1901.04820.
18. Ebrahimi, J.; Fragouli, C. Algebraic Algorithms for Vector Network Coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 996–1007. [CrossRef]
19. Koetter, R.; Medard, M. An algebraic approach to network coding. *IEEE/ACM Trans. Netw.* **2003**, *11*, 782–795. [CrossRef]
20. Jaggi, S.; Sanders, P.; Chou, P.A.; Effros, M.; Egner, S.; Jain, K.; Tolhuizen, L.M.G.M. Polynomial time algorithms for multicast network code construction. *IEEE Trans. Inf. Theory* **2005**, *51*, 1973–1982. [CrossRef]
21. Fragouli, C.; Soljanin, E. Information flow decomposition for network coding. *IEEE Trans. Inf. Theory* **2006**, *52*, 829–848. [CrossRef]
22. Médard, M.; Effros, M.; Karger, D.; Ho, T. On coding for non-multicast networks. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, Monticello, IL, USA, 23–25 September 1998; Volume 41, pp. 21–29.
23. Das, N.; Rai, B.K. On the Message Dimensions of Vector Linearly Solvable Networks. *IEEE Commun. Lett.* **2016**, *20*, 1701–1704. [CrossRef]
24. Sun, Q.T.; Yang, X.; Long, K.; Yin, X.; Li, Z. On Vector Linear Solvability of Multicast Networks. *IEEE Trans. Commun.* **2016**, *64*, 5096–5107. [CrossRef]
25. Etzion, T.; Wachter-Zeh, A. Vector Network Coding Based on Subspace Codes Outperforms Scalar Linear Network Coding. *IEEE Trans. Inf. Theory* **2018**, *64*, 2460–2473. [CrossRef]
26. Sun, Q.T.; Yin, X.; Li, Z.; Long, K. Multicast Network Coding and Field Sizes. *IEEE Trans. Inf. Theory* **2015**, *61*, 6182–6191. [CrossRef]
27. Chou, P.A.; Wu, Y.; Jain, K. Practical network coding. In Proceedings of the Annual Allerton Conference on Communication Control and Computing, Monticello, IL, USA, 23–25 September 1998; Volume 41, pp. 40–49.
28. Li, Y.; Soljanin, E.; Spasojevic, P. Effects of the Generation Size and Overlap on Throughput and Complexity in Randomized Linear Network Coding. *IEEE Trans. Inf. Theory* **2011**, *57*, 1111–1123. [CrossRef]

29. Thibault, J.; Yousefi, S.; Chan, W. Throughput Performance of Generation-Based Network Coding. In Proceedings of the 10th Canadian Workshop on Information Theory (CWIT), Edmonton, AL, Canada, 6–8 June 2007; pp. 89–92. [CrossRef]
30. Trullols-Cruces, O.; Barcelo-Ordinas, J.M.; Fiore, M. Exact Decoding Probability Under Random Linear Network Coding. *IEEE Commun. Lett.* **2011**, *15*, 67–69. [CrossRef]
31. Larsson, P. Multicast Multiuser ARQ. In Proceedings of the IEEE Wireless Communications and Networking Conference, Las Vegas, NV, USA, 30 March–3 April 2008; pp. 1985–1990. [CrossRef]
32. Qureshi, J.; Foh, C.H.; Cai, J. Optimal solution for the index coding problem using network coding over GF(2). In Proceedings of the 9th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks (SECON), Seoul, Korea, 18–21 June, 2012; pp. 209–217. [CrossRef]
33. Lucani, D.E.; Pedersen, M.V.; Ruano, D.; Sørensen, C.W.; Fitzek, F.H.P.; Heide, J.; Geil, O.; Nguyen, V.; Reisslein, M. Fulcrum: Flexible Network Coding for Heterogeneous Devices. *IEEE Access* **2018**, *6*, 77890–77910. [CrossRef]
34. Skevakis, E.; Lambadaris, I. Decoding and file transfer delay balancing in network coding broadcast. In Proceedings of the IEEE International Conference on Communications (ICC), Kuala Lumpur, Malaysia, 23–27 May 2016; pp. 1–7. [CrossRef]
35. Lucani, D.E.; Medard, M.; Stojanovic, M. On Coding for Delay—Network Coding for Time-Division Duplexing. *IEEE Trans. Inf. Theory* **2012**, *58*, 2330–2348. [CrossRef]
36. Sagduyu, Y.E.; Georgiadis, L.; Tassiulas, L.; Ephremides, A. Capacity and Stable Throughput Regions for the Broadcast Erasure Channel With Feedback: An Unusual Union. *IEEE Trans. Inf. Theory* **2013**, *59*, 2841–2862. [CrossRef]
37. Cocco, G.; de Cola, T.; Berioli, M. Performance analysis of queueing systems with systematic packet-level coding. In Proceedings of the IEEE International Conference on Communications (ICC), London, UK, 8–12 June 2015; pp. 4524–4529. [CrossRef]
38. Chatzigeorgiou, I.; Tassi, A. Decoding Delay Performance of Random Linear Network Coding for Broadcast. *IEEE Trans. Veh. Technol.* **2017**, *66*, 7050–7060. [CrossRef]
39. Aljohani, A.J.; Ng, S.X.; Hanzo, L. Distributed Source Coding and Its Applications in Relaying-Based Transmission. *IEEE Access* **2016**, *4*, 1940–1970. [CrossRef]
40. Giacaglia, G.; Shi, X.; Kim, M.; Lucani, D.E.; Médard, M. Systematic network coding with the aid of a full-duplex relay. In Proceedings of the IEEE International Conference on Communications (ICC), Budapest, Hungary, 9–13 June 2013; pp. 3312–3317. [CrossRef]
41. Wunderlich, S.; Gabriel, F.; Pandi, S.; Fitzek, F.H.P.; Reisslein, M. Caterpillar RLNC (CRLNC): A Practical Finite Sliding Window RLNC Approach. *IEEE Access* **2017**, *5*, 20183–20197. [CrossRef]
42. Pandi, S.; Gabriel, F.; Cabrera, J.A.; Wunderlich, S.; Reisslein, M.; Fitzek, F.H.P. PACE: Redundancy Engineering in RLNC for Low-Latency Communication. *IEEE Access* **2017**, *5*, 20477–20493. [CrossRef]
43. Sundararajan, J.K.; Shah, D.; Médard, M.; Sadeghi, P. Feedback-Based Online Network Coding. *IEEE Trans. Inf. Theory* **2017**, *63*, 6628–6649. [CrossRef]
44. Chi, K.; Jiang, X.; Horiguchi, S. Network coding-based reliable multicast in wireless networks. *Comput. Netw.* **2010**, *54*, 1823–1836. [CrossRef]
45. Fragouli, C.; Lun, D.; Medard, M.; Pakzad, P. On Feedback for Network Coding. In Proceedings of the 41st Annual Conference on Information Sciences and Systems, Baltimore, MD, USA, 14–16 March 2007; pp. 248–252. [CrossRef]
46. Keller, L.; Drinea, E.; Fragouli, C. Online Broadcasting with Network Coding. In Proceedings of the Fourth Workshop on Network Coding, Theory and Applications, Hong Kong, China, 3–4 January 2008; pp. 1–6. [CrossRef]
47. Sundararajan, J.K.; Shah, D.; Medard, M. ARQ for network coding. In Proceedings of 2008 IEEE International Symposium on Information Theory, Toronto, ON, Canada, 6–11 July 2008; pp. 1651–1655. [CrossRef]
48. Lucani, D.E.; Medard, M.; Stojanovic, M. Online Network Coding for Time-Division Duplexing. In Proceedings of the IEEE Global Telecommunications Conference GLOBECOM 2010, Miami, FL, USA, 6–10 December 2010; pp. 1–6. [CrossRef]

49. Fu, A.; Sadeghi, P.; Médard, M. Delivery delay analysis of network coded wireless broadcast schemes. In Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC), Paris, France, 1 April 2012; pp. 2236–2241. [CrossRef]
50. Barros, J.; Costa, R.A.; Munaretto, D.; Widmer, J. Effective Delay Control in Online Network Coding. In Proceedings of the IEEE INFOCOM 2009, Rio de Janeiro, Brazil, 19–25 April 2009; pp. 208–216. [CrossRef]
51. Karim, M.S.; Sadeghi, P. Decoding delay reduction in broadcast erasure channels with memory for network coding. In Proceedings of the IEEE 23rd International Symposium on Personal, Indoor and Mobile Radio Communications—(PIMRC), Sydney, Australia, 9–12 September 2012; pp. 60–65. [CrossRef]
52. Sadeghi, P.; Traskov, D.; Koetter, R. Adaptive network coding for broadcast channels. In Proceedings of the Workshop on Network Coding, Theory, and Applications, Lausanne, Switzerland, 15–16 June 2009; pp. 80–85. [CrossRef]
53. Sundararajan, J.K.; Sadeghi, P.; Medard, M. A feedback-based adaptive broadcast coding scheme for reducing in-order delivery delay. In Proceedings of the Workshop on Network Coding, Theory, and Applications, Lausanne, Switzerland, 15–16 June 2009; pp. 1–6. [CrossRef]
54. Fu, A.; Sadeghi, P.; Médard, M. Dynamic Rate Adaptation for Improved Throughput and Delay in Wireless Network Coded Broadcast. *IEEE/ACM Trans. Netw.* **2014**, *22*, 1715–1728. [CrossRef]
55. Wu, F.; Sun, Y.; Chen, L.; Xu, J.; Srinivasan, K.; Shroff, N.B. High Throughput Low Delay Wireless Multicast via Multi-Channel Moving Window Codes. In Proceedings of the IEEE INFOCOM 2018—IEEE Conference on Computer Communications, Honolulu, HI, USA, 15–19 April 2018; pp. 2267–2275. [CrossRef]
56. Garrido, P.; Leith, D.; Aguero, R. Joint scheduling and coding over lossy paths with delayed feedback. *arXiv* **2018**, arXiv:1804.04921.
57. Wu, F.; Sun, Y.; Yang, Y.; Srinivasan, K.; Shroff, N.B. Constant-Delay and Constant-Feedback Moving Window Network Coding for Wireless Multicast: Design and Asymptotic Analysis. *IEEE J. Sel. Areas Commun.* **2015**, *33*, 127–140. [CrossRef]
58. Ho, T.; Viswanathan, H. Dynamic Algorithms for Multicast With Intra-Session Network Coding. *IEEE Trans. Inf. Theory* **2009**, *55*, 797–815. [CrossRef]
59. Eryilmaz, A.; Lun, D.S. Control for inter-session network coding. In Proceedings of the 3rd Workshop on Network Coding, Theory & Applications, Citeseer, San Diego, CA, USA, 29 January 2007; pp. 1–9.
60. Sundararajan, J.K.; Shah, D.; Medard, M. Online network coding for optimal throughput and delay—The three-receiver case. In Proceedings of the International Symposium on Information Theory and Its Applications, Auckland, New Zealand, 7–10 December 2008; pp. 1–6. [CrossRef]
61. Giri, S.; Roy, R. Packet Dropping and Decoding Statistics for Linear Network Coded Broadcast with Feedback. In Proceedings of the IEEE 43rd Conference on Local Computer Networks Workshops (LCN Workshops), Chicago, IL, USA, 1–4 October 2018; pp. 70–77. [CrossRef]
62. Andrews, G.E.; Eriksson, K. *Integer Partitions*; Cambridge University Press: Cambridge, UK, 2004.
63. Cover, T.M.; Thomas, J.A. *Elements of Information Theory*; John Wiley & Sons: Hoboken, NJ, USA, 2012.
64. Li, D.; Sun, X. *Nonlinear Integer Programming*; Springer Science & Business Media: Berlin, Germany, 2006; Volume 84.
65. Lucani, D.E. Composite Extension Fields for (Network) Coding: Designs and Opportunities. In Proceedings of the Network Coding and Designs (Final Conference of COST Action IC1104), Dubrovnik, Croatia, 4–8 April 2016.



© 2019 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (http://creativecommons.org/licenses/by/4.0/).