

# Instantly Decodable Network Coding Protocols with Unequal Error Protection

Muhammad Muhammad, Matteo Berioli, Gianluigi Liva  
 German Aerospace Center (DLR)

Institute of Communications and Navigation  
 Oberpfaffenhofen, Germany

Email: {Muhammad.Muhammad, Matteo.Berioli, Gianluigi.Liva}@dlr.de

Giovanni Giambene  
 CNIT - University of Siena

Department of Information Engineering  
 Siena, Italy

Email: Giambene@unisi.it

**Abstract**—This work aims at introducing two novel packet retransmission techniques for reliable multicast in the framework of Instantly Decodable Network Coding (IDNC). These methods are suitable for order- and delay-sensitive applications, where some information is of high importance for an earlier gain at the receiver’s side. We introduce hence an Unequal Error Protection (UEP) scheme, showing by simulations that the Quality of Experience (QoE) for the end-users is improved even without complex encoding and decoding.

## I. INTRODUCTION

Broadcasting is a technique to deliver information to the mass public, whereas multicasting is to bring this data to a specific group of end users. In this context, satellites help in wide spreading the communication’s geographical area in order to reach larger population, for example thanks to the Digital Video Broadcast - Satellite (DVB-S2) standard. Due to the implementation of Adaptive Coding and Modulation (ACM) schemes, packet erasure rate (PER) is in the order of  $10^{-7}$  in DVB-S2 systems. However, when satellite broadcasting is used to serve mobile users, the terminals suffer from packet losses, because of fading or shadowing. Further, for reliable transmission, lost information has to be retransmitted to the intended users.

The Automatic Repeat reQuest (ARQ) protocol [1] is a very simple solution against packet erasures. Nevertheless, implementing such protocols in a multicast communication system leads to performance degradation, particularly when uncorrelated (i.e., independent) packet losses occur at the receiver side with each receiver sending acknowledgments (ACKs). This low performance is due to the large number of packets retransmission that will take place when larger number of users is connected.

Let us consider a simple scenario, depicted in Figure 1, where three users are being served by the gateway through a satellite that is broadcasting packets to the users. Correctly-received packets are grey ones; instead, missing packets are darker ones. Users will signal their respective packet losses through a return channel to the gateway. With simple ARQ mechanisms, like Selective Repeat ARQ (SR-ARQ), there will be a total of three different packet retransmissions, namely,  $\{P_1, P_3, P_5\}$  in this scenario, for instance. However, for a system with larger set of users, this technique is not suitable

due to the large number of retransmissions and the feedback implosion problem.

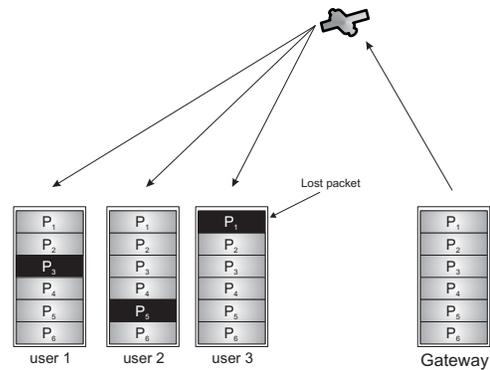


Fig. 1. Scenario under consideration

In order to overcome this problem, efficient alternatives to ARQ protocols for multicast and broadcast were proposed in [2]–[4], based on the idea of erasure recovery. Furthermore, network coding [5] and index coding [6] concepts opened the door for several other algorithms, like the ones considered in [7]–[9]. Another set of methods, based on fountain codes [10]–[12], were introduced. For these mechanisms, unlimited number of redundant packets are generated by the sender for a set of  $K$  source packets, where redundant packets are linear combinations of the source packets, for instance, using Random Linear Network Coding (RLNC). When a receiver succeeds in recovering the  $K$  source packets, it will signal a decoding success to the sender that will stop producing redundancy once all receivers have successfully decoded their intended packets. This requires that every receiver has to solve a system of linear equations (that is built from the successfully-received packets, which are linearly independent) with  $K$  unknowns. Hence, the complex decoding, using Gaussian elimination methods of complexity  $\mathcal{O}(K^3)$ <sup>1</sup>, may consume processing resources at the destination. These protocols require just a signalling feedback (no complete feedback about lost

<sup>1</sup>Smart Gaussian elimination methods can be used to decode fountain codes by exploiting the sparseness of the system of equations. However, the final step of the algorithm turns the solution into a dense system of equations with  $e$  unknowns (where  $e$  is a fraction of  $K$ ).

packets) to tell the sender to stop sending combinations, else it will keep on generating parity packets.

On the other hand, feedback-oriented retransmission techniques were developed, like in [13]–[18]. The authors of [13] have developed an algorithm that finds a low-complexity compromise between the above presented solutions. Their proposed Coded ARQ (C-ARQ) technique excludes some packets from retransmission with the help of a taboo maker that, additionally, helps in choosing the packets for combination. However, C-ARQ may produce linear combinations with more than one unknown (i.e., lost packets by a user), forcing the receiver to wait for a block of codewords before being able to start decoding by means of back-substitution algorithms, for which their complexity scales as  $\mathcal{O}(K^2)$ .

Although coding across information packets at the sender side will increase throughput, the delay will also increase as the receiver has to collect a block of coded packets in order to start decoding. This aspect could not be a problem for some applications that are insensitive to delay, such as file download. Such applications can also be order-insensitive to packet delivery. However, we could consider other examples of multicast applications where the delay performance is important. For instance, let us consider a progressive decoder, which incrementally decodes and renders portions of an image before the entire image has finished downloading. The user's experience is greatly improved using this technique, while downloading the image from the Internet, since a user can preview the available data without having to wait for the entire download to start decoding.

In spite of the great effort done by the scientific community in the field of network coding, all the previously-mentioned techniques propose simple-to-moderate coding operations at the sender side, whereas the receivers have to perform the complex decoding operations. Instantly Decodable Network Coding (IDNC) protocols for broadcast/multicast were first addressed in [7]. IDNC protocols provide instant packet recovery upon the reception of an appropriate IDNC packet. Additionally, IDNC encoding method is implemented using binary XOR. [14]–[16] propose algorithms that target delay-sensitive applications, such as video streaming. Adaptive IDNC algorithms were shown in [17], [18], however, these methods require feedback for every (re)transmitted packet to determine the reception status for every receiver and, hence, make a decision on the packets combination. This technique is not suitable for networks with long Round-Trip Times (RTTs) delay, such as satellite systems.

In this light, the aim of this work is to present two routines, which by approaching the problem from different dimensions try to select the packets for combination. For packets selection, the first algorithm look into the information about the successfully-received packets, whereas the second one selects the packets based on their respective requests from the users. Additionally, these algorithms are designed in a way to make use of high priority (HP) packets in order to increase the user's Quality of Experience (QoE). The QoE for the users is improved by reducing the delay of the HP

packets, but not at the expense of increasing the overall delay of the requested object delivery. Most of the work done so far on Unequal Error Protection (UEP) in the sense of increasing the QoE for users was for rateless codes such as fountain codes and RLNC for multimedia communications, see [19], [20]. The set of HP packets will carry the I-frames in an JPEG image in the progressive decoder example, for instance. The proposed methods belong to the IDNC class, which makes them appropriate for simple-to-design and cost-efficient handheld devices with limited resources.

The paper is organized as follows. The system model is described in Section II. In Section III, the two algorithms are introduced with examples for better clarity. Performance evaluation is provided in Section IV. Finally, conclusions will follow in Section V.

## II. SYSTEM MODEL

Let us assume that a service provider wants to deliver a data file of a certain size to a plurality of users via a multicast channel. These users are using satellite terminals with limited resources, i.e., they cannot perform complex arithmetical operations.

Figure 1 shows the satellite scenario under consideration. The file to be transmitted is fragmented at the gateway into multiple packets of fixed size  $L$  bits. Let us denote the set of source packets by  $\rho = \{P_1, P_2, \dots, P_K\}$ , where  $|\rho| = K$ . Let us assume that there exists a predefined set  $\mathfrak{P} \subset \rho$ , which consists of the HP packets, with a given size of  $|\mathfrak{P}| = K_{\mathfrak{P}}$ . The gateway will multicast the  $K$  source packets to a set of  $N$  users through a satellite. Each user is experiencing independent packet erasures from the others, where the packet error probability  $\varepsilon$  is assumed to be the same for every user. In case of packet erasures, requests for retransmissions are signaled via a reliable feedback channel using Selective Negative Acknowledgments (SNACKs). A SNACK will contain a list of packet indices lost by user  $j$  denoted by  $\phi_j$ . The retransmissions for packet loss recovery starts after collecting the information about the lost packets of all users. Several recovery rounds are considered, until the file is delivered to all the users. Hence, the packets of the file are first transmitted un-coded, but coding of source packets is allowed in the subsequent recovery rounds. From a SNACK (i.e.,  $\phi_j$ ), the sender can deduce the list of packets successfully received by every user, noted as  $v_j = \rho \setminus \phi_j$ . Eventually, the indices of all the lost data packets by every user are signalled in the SNACK at the end of the file transmission and every recovery round. This feedback is assumed to be immediate from all the users.

**Definition II.1.** *A transmitted packet or combination is noted as innovative for user  $j$  if he is missing the packet or one packet from the combination [17].*

**Definition II.2.** *A transmitted combination is noted as instantly decodable for user  $j$  if it includes no more than a single packet that user  $j$  did not receive/decode yet [17].*

**Definition II.3.** *User  $j$  will experience a delay of 1 unit if it*

receives a non-innovative or not instantly decodable packet, see [7].

**Definition II.4.** The completion delay is referred to as the total number of transmissions required for all users in the system to obtain all packets.

**Definition II.5.** The completion delay for the HP packets is defined as the total number of transmissions required for all users in the system to obtain the set of HP packets, i.e.,  $\mathfrak{P}$ .

In the following, we will assume packet combinations obtained by the XOR of at most two source packets.

### III. PACKET SELECTION ALGORITHMS

After the reception of the SNACKs from all users, the packets are grouped in four sets:

- The set  $\mathcal{A}$ , of size  $|\mathcal{A}| = K_{\mathcal{A}}$ , is the set of the packets that have been received by all users.
- The set  $\mathcal{P}$ , of size  $|\mathcal{P}| = K_{\mathcal{P}}$ , is the set of HP packets that have not been received by none or at least one user.
- The set  $\mathcal{U}$ , of size  $|\mathcal{U}| = K_{\mathcal{U}}$ , is the set of low priority (LP) packets that have been received by at most one user. These packets will be denoted as urgent, from now on.
- The set  $\mathcal{Q}$ , of size  $|\mathcal{Q}| = K_{\mathcal{Q}}$ , is the set of LP packets, that have been received by at least two users.

Clearly,  $\mathcal{A} \cup \mathcal{P} \cup \mathcal{U} \cup \mathcal{Q} = \rho$ .

A heuristic approach to provide an improved QoE to the users is to prioritize retransmissions as follows. First, retransmissions shall target a fast delivery of the packets in  $\mathcal{P}$ . This can be obtained by judiciously building linear combinations of the packets in  $\mathcal{P}$ . Then, among the LP packets, those marked as urgent may be addressed first when combining packets. In fact, urgent packets are those missed by most of the users. The eventual fast delivery of the packets in  $\mathcal{U}$  shall thus improve the QoE of a large population of users. Finally, the packets of  $\mathcal{Q}$  can be selected for linear combinations. Having said that, the question is: how to do it? Let us assume that we have an algorithm that takes a set of packets as an input and produces linear combinations. Then, the input of this algorithm could be any element of the following input set:

$$\mathfrak{I} = \{\mathcal{P}, \mathcal{P} \cup \mathcal{U}, \mathcal{P} \cup \mathcal{Q}, \mathcal{Q} \cup \mathcal{U}, \mathcal{Q}\}.$$

Once the four sets of packets, namely,  $\mathcal{P}$ ,  $\mathcal{U}$ ,  $\mathcal{Q}$ , and  $\mathcal{A}$ , are defined, a recovery algorithm is adopted. We describe below two alternatives.

#### A. Algorithm 1: Least Common Users

Algorithm 1 uses information regarding the successfully-received packets by every user, i.e.,  $v_j$ .

**Definition III.1.** Let  $U_{P_i}$  denotes the set of users who successfully-received packet  $P_i$ .

**Rule III.2.** An instantly-decodable packet can only be generated from any two source packets ( $P_i, P_j$ ), with ( $i \neq j$ ), that have been received by two sets of users ( $U_{P_i}, U_{P_j}$ ); where

$U_{P_i} \cup U_{P_j}$  should span all the  $N$  receivers. If ( $U_{P_i} \cup U_{P_j}$ ) does not span all receivers, then  $P_j$  cannot be selected.

#### Algorithm 1

- 1) Select a set of packets from  $\mathfrak{I}$ , say  $\mathcal{X}$ .
- 2) If  $|\mathcal{X}| > 1$ , choose packet  $P_i \in \mathcal{X}$  at random and check the list of users  $U_{P_i}$  who received it, and continue as follows. Otherwise, go to Step 1.
- 3) For every other packet  $P_j \in \mathcal{X}$  ( $i \neq j$ ), determine the set of users ( $U_{P_j}$ ) that received it.
- 4) Choose a packet  $P_j \in \mathcal{X}$  such that the number of elements in  $\{U_{P_i} \cap U_{P_j}\}$  is minimized, i.e.,  $P_j = \underset{P_j \in \mathcal{X}}{\operatorname{argmin}} \{ |U_{P_i} \cap U_{P_j}| \}$ .<sup>2</sup>
- 5) Send  $P_i \oplus P_j$  and remove  $P_i$  and  $P_j$  from  $\mathcal{X}$ . Go to Step 2.

Un-coded retransmitted packets are generated based on Rule III.3. Moreover, the importance of the order of the sets in  $\mathfrak{I}$  is discussed in details later on.

**Rule III.3.** If, after processing the whole set  $\mathcal{P} \cup \mathcal{U}$ , no further possible combinations can be created and  $\mathcal{P} \neq \emptyset$ , then send all the remaining packets  $P_i \in \mathcal{P}$  without coding. The same applies for  $\mathcal{U}$  and  $\mathcal{Q}$  at the end of the set  $\mathcal{Q}$ .

The rationale behind Step 4, in Algorithm 1, is to minimize the delay (as defined in Definition II.3) per user to receive an object. Assume a user  $v_x$  has the packets  $P_1$  and  $P_2$ ; sending him  $P_1 \oplus P_2$  (with  $\oplus$  being binary XOR operation) will not bring him new information (non-innovative packet); therefore, a delay of 1 unit is counted. On the other hand, by receiving  $P_1 \oplus P_3$ , he can decode  $P_3$  and no delay is realized in this case, since this combination reveals new information.

1) *Example 1:* A file is segmented into  $K = 10$  packets ( $\rho = \{P_1, P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9, P_{10}\}$ ) of a fixed size at the gateway to be broadcast to a set of  $N = 4$  users through a satellite. The set of UEP packets is fixed to  $\mathcal{P} = \{P_1, P_5, P_{10}\}$ . The lost packets, i.e.,  $\phi_j$ , are requested from the sender by every receiver via a reliable feedback channel through SNACK messages; we can consider the following cases:

$$\begin{aligned} \phi_1 &= \{P_2, P_3, P_4, P_6, P_7\} \\ \phi_2 &= \{P_1, P_2, P_3, P_5, P_6, P_7\} \\ \phi_3 &= \{P_2, P_4, P_9, P_{10}\} \\ \phi_4 &= \{P_1, P_2, P_3, P_4, P_7, P_9\} \end{aligned}$$

The gateway can thus determine the set of successfully-received packets for each user:

$$\begin{aligned} v_1 &= \{P_1, P_5, P_8, P_9, P_{10}\} \\ v_2 &= \{P_4, P_8, P_9, P_{10}\} \\ v_3 &= \{P_1, P_3, P_5, P_6, P_7, P_8\} \\ v_4 &= \{P_5, P_6, P_8, P_{10}\} \end{aligned}$$

The gateway will post-process the received SNACKs and identify the set of urgent packets as  $\mathcal{U} = \{P_2, P_3, P_4, P_7\}$ , and the set of the packets that have been received by all users  $\mathcal{A} = \{P_8\}$ , whereas the set of the LP other packets is determined as  $\mathcal{Q} = \{P_6, P_9\}$ .

<sup>2</sup>Taking Rule III.2 as granted, this is to minimize the delay.

As it can be clearly realized, if the input of the algorithm was according to the order from  $\mathcal{J}$ , then (taking only  $\mathcal{P}$  at the beginning) the first retransmission could be  $P_1 \oplus P_{10}$ , since  $P_1 \oplus P_5$  does not span the complete set of users (because user 2 is missing from the two sets of users who received the two packets, i.e.,  $U_{P_1}$  and  $U_{P_5}$ , respectively; Rule III.2 is not satisfied), and  $P_5 \oplus P_{10}$  will result in a higher delay. This transmission will be followed by  $P_5 \oplus P_4$  from  $\mathcal{P} \cup \mathcal{U}$ ,  $P_6 \oplus P_9$  from  $\mathcal{Q}$ , and finally  $P_2, P_3, P_7$  are sent without coding.

### B. Algorithm 2: Highly Demanded Packet

We propose an algorithm in order to reduce the number of packets to be chosen for a combination. The sets from  $\mathcal{J}$  are also required in the predefined order.

The sets  $\mathcal{A}, \mathcal{P}, \mathcal{U}, \mathcal{Q}$  as well as the list of indices  $\phi_j$  of the packets lost by the  $j$ -th user can be conveniently displayed through a bipartite graphs, where we associate a user node to each user, and a packet node to each packet. Thus, the graph will possess  $K$  packet nodes and  $N$  user nodes. The  $i$ -th packet node is connected to the  $j$ -th user node if  $P_i$  was not received by the  $j$ -th user, i.e., if  $P_i \in \phi_j$ .

An example of the graph is depicted in Figure 2, where we further clustered packet nodes according to sets  $\mathcal{A}, \mathcal{P}, \mathcal{U}, \mathcal{Q}$ . Clearly, the nodes associated with packets in  $\mathcal{A}$  are not connected to the user nodes. Moreover, already at moderate packet loss rates (e.g.  $\varepsilon \simeq 10^{-2}$ ), the graph will be sparse, i.e., a few edges will emanate from each user node. More specifically, if we denote by  $d_j$  the number of edges connected to the user node  $v_j$ , we will have that  $\mathbb{E}[d_j] = \varepsilon K \ll K$ . We denote the number of edges connected to a node (either user or packet node) as the node degree.

#### Algorithm 2

- 1) Select a set of packets from  $\mathcal{J}$ , say  $\mathcal{X}$ .
- 2) If  $\mathcal{X} \neq \emptyset$ , choose a packet node  $P_i \in \mathcal{X}$ . Otherwise, another set is required, go to Step 1.
- 3) Once a packet node  $P_i$  has been selected, all its connected user nodes will be marked as deactivated.
- 4) Moreover, for all deactivated user nodes, their neighboring (i.e., connected) packet nodes are placed in idle state.
- 5) Copy the set of non-deactivated packet nodes in  $F$ . If  $F = \emptyset$ , go to Step 2. Otherwise, continue as follows.
- 6) For every packet node  $P_j \in F$  determine the number of edges it holds (i.e., its degree) ( $d_j$ ).
- 7) Choose a packet node  $P_j \in F$  such that 
$$P_j = \max_{P_j \in F} \{d_j\}.$$
- 8) Send  $P_i \oplus P_j$  and remove packets  $P_i$  and  $P_j$  from  $\mathcal{X}$ . Delete  $F$  as well. Go to Step 2.

Un-coded retransmitted packets are generated based on Rule III.3.

1) *Example 2:* Using the sets  $\phi_j$  for every user, a bipartite graph can be built, and it will be easier to identify the packets to be combined. Figure 2, for instance, shows the case of having the four sets (from Example 1) with information about packet requests from  $\phi_j$ .

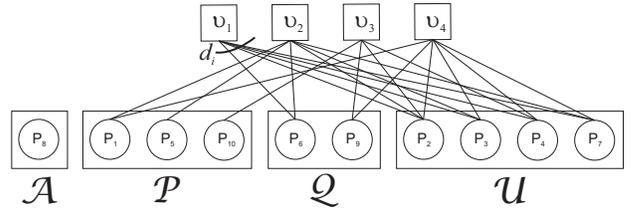


Fig. 2. Bipartite graph visualization of Example 2

As it can be clearly realized, taking only  $\mathcal{P}$  at the beginning (as in  $\mathcal{J}$ ), when  $P_1$  is selected,  $P_5$  will be deactivated, because it was also requested by user 2. Only  $P_{10}$  is still available and it can be combined with  $P_1$  generating  $(P_1 \oplus P_{10})$ . The other packets are generated in the same manner following the order of sets presented earlier in  $\mathcal{J}$ . Finally, the complete set of recovery packets will be similar to those in Example 1.

### C. The Order of Sets in $\mathcal{J}$

The goal of the two algorithms (1 and 2) is to present IDNC techniques with high priority packets that increase the QoE for the users. Hence, the order of the sets of packets in  $\mathcal{J}$  plays a vital role for increasing the QoE, i.e., reducing the delay of the HP packets, minimizing the delay (from Definition II.3) experienced by a user, and keeping a total completion time (Definition II.4) as low as possible.

Eventually, trying to find first the combinations only from  $\mathcal{P}$ , then from  $\mathcal{P} \cup \mathcal{U}$  and  $\mathcal{P} \cup \mathcal{Q}$  will reduce the delay and both the total completion time and the HP delay. For instance, back to Example 1; if  $\mathcal{P} \cup \mathcal{U}$  was considered first, then  $P_{10} \oplus P_3$  will be generated. Although this combination reduces the delay per user on average; however, the HP delay will increase, since instead of using one combination to send  $P_1$  and  $P_{10}$  (as in the examples) now two combinations are required (one for  $P_1$  and another for  $P_{10}$ ).

## IV. PERFORMANCE EVALUATION

In this section, we will evaluate the performance of the proposed algorithms with simulations and compare them to some other protocols from the literature, such as SR-ARQ and opportunistic random coding, in terms of average completion time and median delay. The simulations were run 100 times and the shown results are the average over all these runs.

The opportunistic algorithm is taken from [7]. However, since the *order* of the packets to be chosen for the combination is not defined, we have chosen a random packet, for which the resulting combination should be decodable by all users.

### A. Simulation Results: Median Delay

The median delay is estimated per user. It is calculated according to Definition II.3. The median delay is evaluated over several rounds until the complete delivery of the requested object. The parameters for this test were set as follows;  $K = 100$ , PER of  $\varepsilon = 0.1$ , and the HP packets were forming 20% of the total number of the source packets  $K$ . As it is clearly shown in Figure 3, algorithms 1, 2, and opportunistic

have comparable delay scores, whereas the SR-ARQ shows the highest values. For example, for  $N = 50$  users, the median delay is reduced by 45%, when using the instantly decodable property. This means that the file is delivered within almost half the time that is required by traditional methods, like SR-ARQ.

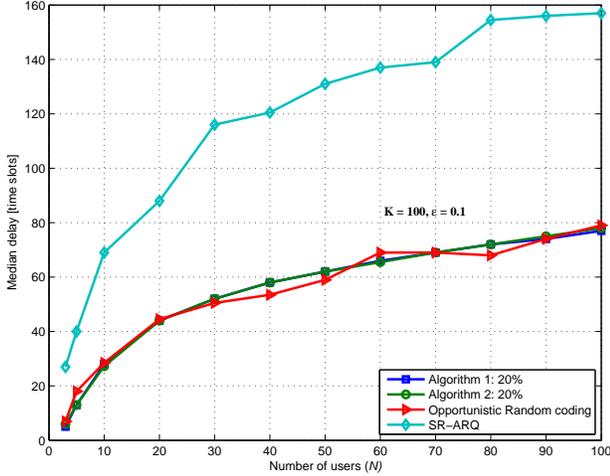


Fig. 3. Median delay vs. number of users

### B. Simulation Results: Expected Completion Time

The average completion time and the average delay of the HP packets was calculated according to the Definitions II.4 and II.5, respectively.

The plot in Figure 4 shows the average completion time as a function of  $N$ . The performance of the two presented algorithms is compared to the opportunistic and the SR-ARQ. As it can be clearly seen, the SR-ARQ has the longest delivery time, whereas algorithms 1 and 2 overlap with the opportunistic. The setup of this experiment included a set of  $K = 100$  source packets and each user was experiencing an independent but equal PER of  $\varepsilon = 0.1$ . An interesting observation can be realized with the delay of the HP packets, which demonstrated 20% of the total size of the source packets, i.e.,  $K_{\mathcal{H}\mathcal{P}} = 0.2K$ . Since these packets are of high priority, the encoder tries to find a suitable combination (according to the order of the sets of packets in  $\mathcal{J}$ ), hence, they show low completion delay. For instance, at  $N = 50$ , 65% of the time is saved to allow the receivers to start gaining something. This will, of course, increase the QoE for the end-user, as he will be able to get some information of the requested object even before the complete download, not as with the other algorithms, such as by the opportunistic random coding, where prioritized packets are not foreseen. Finally, no further delay is expected here, like in block codes, since the codewords are instantly decodable.

Figure 5 plots the average completion time for fixed  $N = 100$ ,  $K = 100$  and variable PER. The plot shows the performance of the two algorithms, which is identical to the opportunistic technique. Even with high PER values, when the

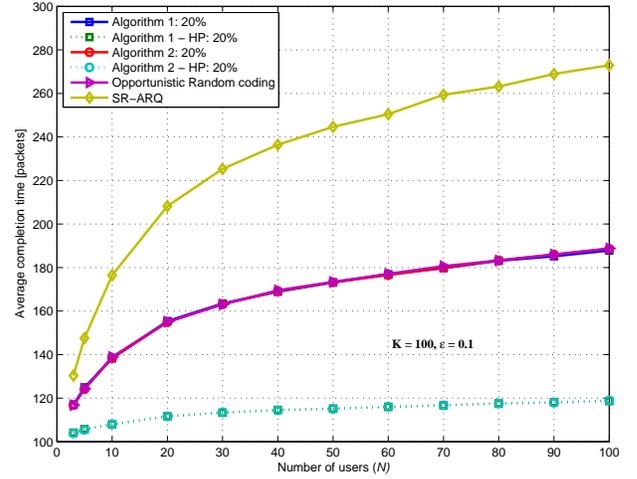


Fig. 4. Average completion time vs. number of users

total completion delay is relatively high, the completion delay of the HP packet is fairly low. This means that the users can still gain in performance instead of having idle time, as with the opportunistic algorithm, for instance.

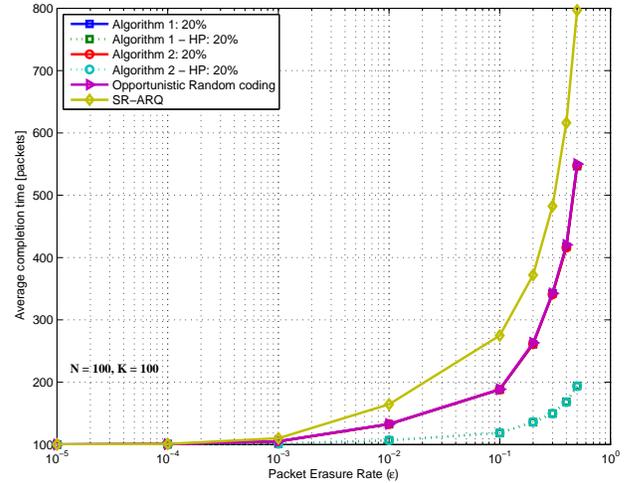


Fig. 5. Average completion time vs. packet erasure rate

The average completion delay is depicted in Figure 6 versus the size of the source packets set ( $K$ ), with a fixed  $N = 100$  receivers and PER  $\varepsilon = 0.1$ . Only Algorithm 1 is shown, since Algorithm 2 has a comparable performance. Further, we show the performance of the algorithm revealing the change of the amount of the high priority packets, i.e.,  $K_{\mathcal{H}\mathcal{P}}$ , with 10, 20, and 50 percent of  $K$ . It is clear that the lower the amount of the HP packets, the lower will be their completion time, but not at the expense of changing the overall completion delay.

## V. CONCLUSION

In this work, we have presented two alternatives for IDNC applied to reliable multicast. These algorithms can handle high priority packets with the lowest possible completion delay, but

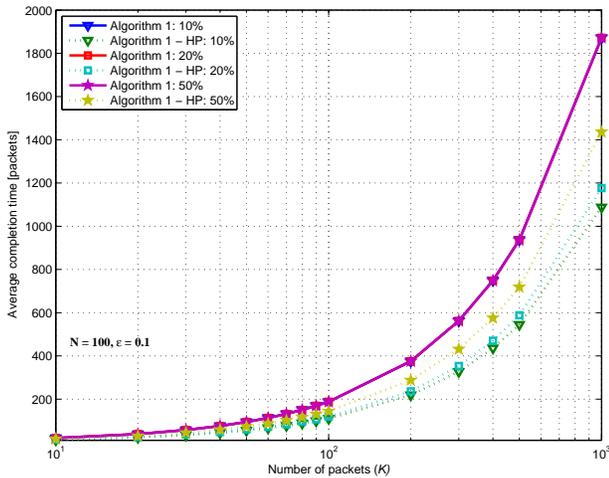


Fig. 6. Average completion time vs. number of packets

not at the expense of increasing the overall delay. Further, the instantly-decodable property is held for all of the generated combinations. It was shown, through intensive simulations, that the QoE for the end-users can be improved without the implementation of complex algorithms neither at sender nor at the receiver side. Finally, although the two algorithms have similar performance, however, as  $K/N \gg 1$ , Algorithm 2 is more suitable for implementation, since the space for searching for the packets to be combined is reduced by eliminating some candidates.

## REFERENCES

- [1] D. Bertsekas and R. Gallager, *Data Networks*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1987.
- [2] J. Metzner, "An Improved Broadcast Retransmission Protocol," *IEEE Transactions on Communications*, vol. 32, no. 6, pp. 679 – 683, Jun. 1984.
- [3] J. Nonnenmacher, E. Biersack, and D. Towsley, "Parity-based Loss Recovery for Reliable Multicast Transmission," *IEEE/ACM Transactions on Networking*, vol. 6, no. 4, pp. 349 –361, Aug. 1998.
- [4] B. Adamson, C. Bormann, M. Handley, and J. Macker, "NACK-Oriented Reliable Multicast Protocol," IETF RFC, Nov. 2004.
- [5] R. Ahlswede, N. Cai, S.-Y. R. Li, and R. W. Yeung, "Network Information Flow," *IEEE Transactions on Information Theory*, vol. 46, no. 4, pp. 1204–1216, Jul. 2000.
- [6] Z. Bar-Yossef, Y. Birk, T. Jayram, and T. Kol, "Index Coding With Side Information," *Information Theory, IEEE Transactions on*, vol. 57, no. 3, pp. 1479 –1494, Mar. 2011.
- [7] L. Keller, E. Drinea, and C. Pragouli, "Online Broadcasting with Network Coding," in *Fourth Workshop on Network Coding, Theory and Applications, 2008. NetCod 2008.*, Hong Kong, Jan. 2008, pp. 1–6.
- [8] J. K. Sundararajan, D. Shah, and M. Medard, "ARQ for Network Coding," in *IEEE International Symposium on Information Theory, 2008. ISIT 2008.*, Toronto, ON, Jul. 2008, pp. 1651–1655.
- [9] A. Tehrani, A. Dimakis, and M. Neely, "Bipartite Index Coding," in *Information Theory Proceedings (ISIT), 2012 IEEE International Symposium on*, Jul. 2012, pp. 2246 –2250.
- [10] J. Byers, M. Luby, M. Mitzenmacher, and A. Rege, "A Digital Fountain Approach to Reliable Distribution of Bulk Data," *SIGCOMM Computer Communications Rev.*, vol. 28, pp. 56–67, October 1998.
- [11] M. Luby, "LT Codes," in *Proceedings of the 43rd Symposium on Foundations of Computer Science*, ser. FOCS '02. Washington, DC, USA: IEEE Computer Society, 2002, pp. 271–.
- [12] A. Shokrollahi, "Raptor Codes," *IEEE/ACM Transactions on Networking*, vol. 14, pp. 2551–2567, Jun. 2006.
- [13] G. Liva, C. Kissling, and C. Hausl, "A Simple Coded ARQ for Satellite Broadcasting," *Journal of Communications and Networks*, vol. 12, no. 6, pp. 577–581, Dec 2010.
- [14] S. Sorour and S. Valaee, "A Network Coded ARQ Protocol for Broadcast Streaming over Hybrid Satellite Systems," in *Personal, Indoor and Mobile Radio Communications, 2009 IEEE 20th International Symposium on*, Sep. 2009, pp. 1098 –1102.
- [15] S. Sorour, S. Valaee, and N. Alagha, "Joint Control of Delay and Packet Drop Rate in Satellite Systems using Network Coding," in *Advanced satellite multimedia systems conference (asms) and the 11th signal processing for space communications workshop (spsc), 2010 5th*, Sep. 2010, pp. 46 –53.
- [16] S. Sorour and S. Valaee, "On Minimizing Broadcast Completion Delay for Instantly Decodable Network Coding," in *Communications (ICC), 2010 IEEE International Conference on*, May 2010, pp. 1 –5.
- [17] P. Sadeghi, D. Traskov, and R. Koetter, "Adaptive Network Coding for Broadcast Channels," in *Network Coding, Theory, and Applications, 2009. NetCod '09. Workshop on*, June 2009, pp. 80 –85.
- [18] P. Sadeghi, R. Shams, and D. Traskov, "An Optimal Adaptive Network Coding Scheme for Minimizing Decoding Delay in Broadcast Erasure Channels," *EURASIP Journal on Wireless Communications and Networking*, vol. 2010, 2010.
- [19] D. Vukobratovic and V. Stankovic, "Unequal Error Protection Random Linear Coding for Multimedia Communications," in *Multimedia Signal Processing (MMSP), 2010 IEEE International Workshop on*, oct. 2010, pp. 280 –285.
- [20] S. Ahmad, R. Hamzaoui, and M. Al-Akaidi, "Unequal Error Protection Using Fountain Codes With Applications to Video Communication," *Multimedia, IEEE Transactions on*, vol. 13, no. 1, pp. 92 –101, feb. 2011.