Abstract—In this paper, multiuser ARQ is extended to multicasting. The core idea is that the sender, based on feedback from users regarding successfully received transmissions, adapts code weights for data packet linear combinations that are then sent. Each user exploits its previously received information in decoding the linearly combined packets. Specifically, a throughput optimal, low en-/decoding complexity enabling, low overhead and on-line multicast coding and scheduling algorithm is devised based on a per user rank increase criterion. For throughput optimality, a minimum field size criterion is derived. Relative previous work, which adaptively identifies sets of users suited to receive linearly combined packets and uses GF(2) and XOR coding, the proposed method adaptively select weights from a sufficient large finite field for optimality instead. Throughput is analyzed and simulated, and en-/decoding complexity, signaling overhead, and latency etc. are studied through realistic simulations. Overall, it is found that the throughput is significantly higher than multicast Selective Repeat ARQ, and that the optimal throughput for an erasure channel is attained.

Keywords- Multicast, ARQ, Network Coding, Cross-layer, Feedback, Multiuser.

I. INTRODUCTION

Reliable and efficient multicasting in wireless networks is of interest for many emerging services like Mobile TV and other potential broadcast orientated services in cellular systems. While automatic repeat request (ARQ) [1] as a method for reliability and lossless communication from its infancy in the mid 1940’s was developed for reliable unicast, various multicast derivatives have been proposed. Several approaches for multicast ARQ have been studied, e.g. based on Go-back-N in [2][3], and a type of Selective Repeat (SR) in [3]. While unicast SR-ARQ has been known to have the highest throughput, though with extra protocol complexity over Go-back-N and Stop-and-Wait ARQ, such approach still performs increasingly poor for multicasting in lossy channels when the number of users $K$ increases. Another idea, explored e.g. in [4]-[7], is the idea to apply forward error correction (FEC) and erasure processing methods to further improve reliable multicast performance. A more recent multicast approach, while not an ARQ method per se, is fountain coding (FC) [8][9] (Tornado and LT) [10] (Raptor). In FC, new pseudo-randomly generated parity information is produced until all users have decoded the sent data file, whereby an ACK is typically issued for each user. In FC, $\epsilon$ extra parity information needs to be sent for a high probability of decoding, giving a throughput of order $T = p/(1+\epsilon)$, where $\epsilon$ becomes reasonable small only for very large number ($=10^3-10^5$) of input packets. To illustrate that the number of input packets $k$ need to be large for a small

Fig. 1. Multicast Multiuser ARQ System Architecture

fixed relative overhead (OH), one may note that the probability of decoding a random binary matrix is upper bounded by $2^{k-n}$ [10] or $2^{-k\epsilon}$ for $n = k(1+\epsilon)$ as the number of output packets.

A different approach, which will be the main focus here, adopts the idea of applying adaptive error correction based on feedback (FB). This was introduced in [11], and assumingly based on this work further explored in [12]. A similar idea has also been considered in Multiuser ARQ (MU-ARQ) [13][14] that studies ARQ for multiple unicast flows. The basic idea is to, through a coding process, exploit users’ previously received packets when retransmitting outstanding packets. It turns out that under wise selection, outstanding and to be retransmitted packets can be merged together by coding, e.g. by bit-wise XORing, into what is here denoted a composite data packet (CP). The users may then use the previously received packets to decode and extract relevant new information from each CP. One difference for (unicast) MU-ARQ is that overhead packets are exploited, which per definition does not exist in [11][12] due to the multicast topology. Briefly, the idea presented in [11] is to first send a number of regular data packets (RPs) and subsequently use a hypergraph coloring based algorithm to find suitable sets of users to which it may send an XOR based CP. Apart from algorithmic details, adopting a matrix and not a hypergraph based approach, [12] operates in a very similar manner.

While network coding (NC) [15] was initially developed to exploit network connectivity over-provisioning rather than the existence of lossy links (which is nevertheless now seen as a fertile and developing ground for extending the NC idea) and did not originally exploit the opportunities with FB, there is clearly a connection here to the NC area through the notion of mixing data. Hence, the insightful idea in [11] should be recognized as an early contributor to the NC field.

In this paper, a new scheduling and coding principle for Multicast ARQ that improves the throughput relative multicast SR-ARQ as well as [11] and [12] is developed. Inspiration is
gained from [13][14] that permits greater flexibility than prior adaptive FB based ARQ work. First, the philosophy will, in contrast to [11][12] where RPs are always sent first before a CP is sent to a set of selected users, be to avoid any strict enforced transmit order. Second, also differing from [11][12], larger finite field sizes than GF(2) is used when needed. Note that it is not the use of a larger field size for coding, e.g. well known from RS codes and random NC [16], that is important here, but rather that it appears as a prerequisite in connection with the number of users, number of packets and the use of FB for optimal throughput.

More specifically, a K-user multicast MU-ARQ scheme, Fig.1, is proposed for which analytical and simulated results are compared to multicast SR-ARQ while assuming primarily identical independently distributed (iid) reception probabilities since the iid assumption (with all links being equally bad) enables the worst case situation in terms of performance, complexity and signaling OH for a fixed number of users to be studied. Nevertheless, the proposed method and algorithm are applicable in non-iid channels. An important finding is that the proposed scheme, provided the code field size meets a certain condition, attains the maximum possible throughput for an erasure channel that does also not degrade with increasing number of users. Note that the method also differs to multicasting based FC and random NC, as it provides deterministic decodeability, and can handle any number of packets (i.e. both few and many) with optimal performance. A further objective here is to provide a MU-ARQ framework, i.e. its generalization to both uni- and multicasting.

The rest of this paper is organized as follows. An introduction and a framework to uni- and multicast MU-ARQ, as well as a description of the proposed algorithm together with a discussion on the minimum required code field size for optimal throughput, is given in II. The performance of multicast based MU-ARQ and SR-ARQ are analyzed in III. Simulation results on throughput, en-/decoding complexity, signaling OH etc are presented in section IV. The paper is subsequently summarized in V.

II. MULTICAST MU-ARQ

A. An introductory example of Uni- and Multicast MU-ARQ

MU-ARQ, in its original form, was developed for multiple parallel unicast flows, i.e. flows originating from one sender with different receiving nodes, but its applicability to multicasting is evident. In this respect, it shares many aspects with [11][12], but it imposes reduced constraints with respect to the packet transmit order, the field size, and the method for adaptation. This may at first sight look like subtle differences, but are in fact vital clues for enhanced throughput performance. We now start with examples of uni- and multicast MU-ARQ.

Fig. 2 illustrates the uni- and multicast operation with an XOR based coding approach, which is later shown to be optimal for two users. Starting with unicast, it is seen that after the first two transmissions, $u_1$ and $u_2$ have not yet received their own (intended) packets, but each others (unintended) packets. The sender then schedule and transmit, based on FB, a bitwise XOR encoded CP, which is received by $u_1$. This enables $u_1$ to extract its desired packet $D_1(1)$. After sending $D_1(2)$, which is only received by $u_2$, and sending a CP $D_1(2) \oplus D_2(1)$ that is received by both users, $u_1$ and $u_2$ can extract their desired packets $D_1(2)$ and $D_2(1)$ respectively. In the multicast case, with the assumption of the same link errors, the CPs are sent at the same instances. Yet, in contrast to unicast, where $u_1$ ends up with the useful packets $D_1(1), D_2(2)$ and $u_2$ ends up with $D_2(2)$, the result for multicast is that both $u_1$ and $u_2$ receives all three packets $D(1), D(2), D(3)$. In general, different scheduling and coding choices will be done for the unicast and multicast case, as different goals are aimed at.

B. Optimal Multicast MU-ARQ

Basic operation, motivation and differences to closely related work: The basic idea behind the algorithm is now introduced with some notes on system modeling first. CPs, used so far, can be seen as linear combinations of one or more (equal sized) RPs. In this respect, any RP are just a special case of a CP, and will in the following be seen as a CP provided nothing else is stated. While the linear combination can in principle be performed with regular arithmetic, causing (in relative terms) low OH carry bits for the CP, it is preferable to use an arithmetic that ensures that the CP length remain the same as the RPs. Hence, arithmetic based on finite fields (for suitable substrings of the packets) is used. Now, assuming that each user has received one or more CPs, one may for each user see this as a system of linear equations where the CPs and the used weights (e.g. inband signaled) are known, but the data packets are unknown and shall be solved for.

The crucial idea for optimal Multicast MU-ARQ is how the adaptation is performed, i.e. at the scheduling and coding instance, the sender forms a linear combination of one or more regular multicast packets into a CP. The weights for the linear combination, which the (substrings of) RPs are multiplied with, are wisely selected based on the FB of previously received CPs such that the rank would always increase for each user’s linear system of equations if a CP with the selected weights were received. Thus, if $N$ data packets are to be sent, each user is deterministically ensured to receive (at least) $N$ CPs that guarantees the ability to form $N$ linearly independent rows from which all data packets can be retrieved from. This is
optimal in the sense that each receiver is guaranteed to retrieve the desired \( N \) data packets from the first \( N \) CPs it receives.

In order to ensure that the rank increases for all users when testing a weight vector, it is in contrast to [11][12] generally not sufficient to use GF(2) based weights. This is for instance easily seen for 3 users, 2 RPs and using GF(2). Assume that if \( u_i \) receives \( D(2) \), \( u_2 \) receives \( D(1) \), and \( u_3 \) receives \( D(1)+D(2) \), then there is no possibility to send a new linear combination of \( D(1) \) and \( D(2) \) with the field such that the rank increases for all users’ weight matrices. However, increasing the field to e.g. GF(3), or larger, enables \( D(1)+2D(2) \) to be sent as it increases the rank for all users’ weight matrices. Based on this example, one could oppose the use of GF(3) or any other non-GF(2\(^m\)) field as this could increased OH, i.e. let’s say that for the GF(3) three digits are inefficiently mapped into two bits. This can be solved by a simple base change, i.e. perform the coding in any desired field and subsequently change the base to binary or any other \( 2^m \) base suitable for modulation and FEC coding. A simpler solution is to always operate in a sufficiently large GF(2\(^m\)) field.

By selecting elements in a weight vector, as described above, another difference in contrast to [11][12] becomes apparent as one does not necessarily always sends RPs first and then tries to find a suitable set of users to which an XOR based CP can be sent to. With other words, and importantly, it is the weights that are adapted, rather than as in [11][12] finding sets of users that may receive an XOR based CP.

System model and practical algorithm: Formalizing the basic scheduling and coding idea gives the following. Each users a priori information, i.e. previously received CPs, can be described as the following linear system of equations

\[
\begin{bmatrix}
    c_{1}^{(k)} \\
    \vdots \\
    c_{M_k}^{(k)}
\end{bmatrix} =
\begin{bmatrix}
    w_{1,1}^{(k)} & \cdots & w_{1,N}^{(k)} \\
    \vdots & \ddots & \vdots \\
    w_{M_k,1}^{(k)} & \cdots & w_{M_k,N}^{(k)}
\end{bmatrix}
\begin{bmatrix}
    d_1 \\
    \vdots \\
    d_N
\end{bmatrix},
\]

where \( d_n \) is the value of the \( n \):th regular packet, \( c_{m}^{(k)} \) is the value for the \( m \):th CP for user \( k \), \( w_{m,n}^{(k)} \in GF(s^k) \) is a weight factor for user \( k \) and regular packet \( n \) and received CP \( m \) by user \( k \) with \( s \) as a prime number and \( v \in \mathbb{N}^+ \), \( M_k \) is the index for the currently last received CP by user \( k \) and \( N \) is the total number of regular packets to be transferred. While \( N \) may be fixed, and one could communicate a mere block of data packets, one may also over time append new data packets and hence increase \( N \). We now write (1) for each user \( k \) as

\[
C_k = W_k D_k.
\]

With this notation, the core scheduler and coding operation is illustrated in Fig. 3, where it is shown that based on the weight matrix \( W_k \), which normally differ among users, a tentative (row) weight vector \( \alpha \) is amended to every \( W_k \) and it is checked that the rank increases for all users that have not yet received all packets. For each user with a successful reception, \( W_k \) is then extended and updated with the used \( \alpha \).

- Consider a multicast group of users, where users are indexed by \( k \).
- With \( N \) regular data packets to multicast
- For each user \( k \), retrieve the weight matrix \( W_k \).
- \( \forall k \) select a code vector \( \alpha \) such that

\[
\text{Rank}(W_k') = \text{Rank}(W_k) + 1,\text{ } \text{where } W_k' \neq [W_k; \alpha]
\]

for those \( k \) that \( \text{Rank}(W_k) < N \).
- Form and send a composite packet based on the code vector
- Update all weight matrices based on feedback of reception

Fig. 3. Core scheduling and coding algorithm

- Generate \( \text{ref}(W_k) \), where \( \text{ref} \) is the reduced row echelon form
- Determine \( R_k = \text{rank}(W_k) \)
- Assign non-ref-reducible non-zero weights to \( \alpha \) at the positions specified by \( R_k + 1 \), with the remaining positions assigned zero, \( s.t \)

\[
\text{rank}(W_k') = \text{rank}(W_k) + 1,\text{ where } W_k' = [W_k; \alpha] \forall k \text{ with } R_k < N
\]

Fig. 4. Practical scheduling and coding algorithm

\[
\begin{bmatrix}
    1 & 1 & 0 & 0 & 0 \\
    0 & 1 & 1 & 0 & 0 \\
    0 & 0 & 0 & 0 & 1
\end{bmatrix}, \quad \begin{bmatrix}
    1 & 0 & 0 & 0 & 0 \\
    0 & 2 & 1 & 0 & 1 \\
    0 & 0 & 1 & 1 & 1
\end{bmatrix}, \quad \begin{bmatrix}
    1 & 0 & 0 & 0 & 0 \\
    0 & 1 & 1 & 0 & 0 \\
    0 & 0 & 0 & 0 & 1
\end{bmatrix}
\]

Fig. 5. Simulated example weight matrices, \( K=3 \) users and \( N=5 \) packets.

Fig. 4 shows a proposed (more practical) algorithm that includes the deterministic rank increase idea. This has been implemented in Matlab and is used for the simulation results in IV. An objective in the algorithm design here has been to strive for low encoding and decoding complexity as well as signaling OH, e.g. in part by considering the actual number of users rather than designing for the worst case with a fixed maximum number of users. Therefore, the algorithm strives to ensure sparse and essentially band-diagonally shaped weight matrices. This is accomplished by that the sender keeps track of a reduced row echelon form for each \( W_k \), and identifies a for all user rank increasing weight vector \( \alpha \) that assigns non-reducible non-zero weights on the \( \text{rank}(W_k+1) \) position for all users. Occasionally, one may need to resort to a heuristic counter like exhaustive search for a weight vector \( \alpha \) with not just all ones at the \( \text{rank}(W_k) + 1 \) positions for all \( K \) users. In this search operation, respective positions are in turn assigned a single weight position of 2, then going through all permutation of two, three, four etc weight positions of weight 2 and so on.

Fig. 5 gives an example of some resulting full rank and essential band-diagonal weight matrices for 3 users, 5 multicast packets, \( iid \) packet erasure channels with reception probability \( p=0.5 \), and the example field \( GF(4) \) for the algorithm.

Condition for optimality: A valid question is if there exists a minimum required field size that ensures that one can always fulfill the rank increase condition for all users, which in turns would ensure the least number of transmissions? Let \( F \) be the finite field under consideration with \( \lambda \) elements. Assume that each of the \( K \) users have received \( N-1 \) CPs, and that the rank for each user is \( N-1 \). Further assume that each CP is only received by a single user, or equivalently that every row for each \( W_k \) of a user \( k \) differs from any other row for \( W_k' \) of user \( k' \). With those two assumptions, we have out of the
$\lambda^{N-1}$ total potential weight vectors, with the all zero word excluded, limited the choices of finding a new (for all users) rank increasing weight vector as much as possible. If we can find at least one weight vector that increases the rank for all users, and the CP based on this weight vector is received by at least one user, this user will have full rank and can be excluded in subsequent rank considerations. If the set of users are diminished, the problem of finding a permissible weight vector is eased, and hence the situation where all users have received all but their last CP is the worst and limiting case. So we have $\lambda^{N}$ possible weight vectors, we remove the all zero word, subtract the number of all possible permutations of $\lambda^{N-1}$ weightings of the $N-1$ rows in $W_k$ with the zero word excluded (as it has been counted already), and we do so for all $K$ users. The remaining number of weight vectors must be at least one to ensure that all users achieve full rank in the end. In all, the condition is $\lambda^{N} - K\lambda^{N-1} + (K-1) \geq 1$ that may be more conveniently written as

$$\lambda^{N} - K\lambda^{N-1} + (K-1) \geq 1 \quad (3)$$

or at closer scrutiny, (3) may for integers be written as

$$\lambda \geq K \quad \text{if } K \geq 2 \text{ and } N \geq 2$$

$$\lambda = 2 \quad \text{Otherwise} \quad (4)$$

Hence, as long as the field size is at least equal to the number of served users, the rank can always increase for all users in each transmission. Practically, one would fix an upper limit of number of users for which optimal performance shall be achievable. Simulations have also verified (3) and (4).

While (3) provides the optimality condition, it is also overly conservative in practice. The reason for this is that a linear combination of two or more weight vectors for one user may be a weight vector used or possibly one that can be constructed from two or more weight vectors for another user. Whether this is the case depends on the exact realization of the $W_k$ s.

A more practical, though not throughput optimal, field size estimation is as follows. From $\lambda^{N-1}$, subtract the number of used weights vectors $K(N-1)$ (remember all weight matrices have $N-1$ rows), and subtract, in a statistical sense, the number of different weight vectors resulting from linear combinations of each users weight vectors, $(\lambda^{N-1})(1-Q^K)$, where $Q = 1 - (\lambda^{N-1} - 1 - (N-1)!)/(\lambda^{N} - 1)$ is one minus the fraction of weight vectors that may be formed by linear combinations of multiple weight vectors for one user, or

$$\lambda^{N} - K(N-1) - (\lambda^{N-1})(1-Q^K) \geq 1, \quad (5)$$

which for large field sizes degenerates to (4). This may for large $N$ be approximated by

$$\lambda^{N}(1 - \lambda^{-1})^{K} \geq K(N-1), \quad (6)$$

Examining (6), one finds that a field size of about $\lambda = 3$ is practically adequate even for small $N$ and reasonable large $K$.

III. THROUGHPUT PERFORMANCE ANALYSIS

The throughput efficiency is examined for $K$ users and iid reception (failure) probabilities, i.e. $p_k = p \ (q_k = 1 - p_k = q)$, by means of analysis and an on-line algorithm based simulation. Further assumptions are non-sequence number limited ARQ, error free FB, and full buffers. We compare two different cases, classical Multicast based SR-ARQ, and the proposed optimal Multicast MU-ARQ method for arbitrary number of $N \in N^*$ packets.

The negative binomial distribution of the discrete r.v. $X$ describes the distribution for the number of $i$ failures before the $N^{th}$ success in a Bernoulli process, with probability $p$ of success on each trial. This is exactly the situation that the algorithm in Fig. 3 and 4 ensures for each user, as it only needs to receive $N$ CPs to decode $N$ packets. The pmf and cdf for $X$, with positive integer number, is given by [17] as

$$f_X(N, p; i) = \binom{N-1}{i} p^N q^{i} \quad (7)$$

and

$$F_X(N, p; i) = I_p(N, i + 1) = \sum_{j=N}^{N+i} \binom{N+i}{j} p^j q^{N+i-j} \quad (8)$$

respectively, where $I_p(...)$ is the regularized incomplete beta function that for positive integer arguments can, as in (8), be expressed in a sum with binomial terms. We are interested to determine the maximum number of failures, and subsequently total number of transmissions, for $K$ users with iid r.v.’s $X_k$, i.e. $Y = \max[X_1, X_2, ..., X_K]$. The resulting cdf for $Y$ is

$$F_Y(K; i) = \left(F_X(i)\right)^K. \quad (9)$$

From the definition of the pmf for a discrete r.v. one gets

$$f_Y(K, N, p; i) = F_Y(K, N, p; i) = F_Y(K, N, p; i-1). \quad (10)$$

that allows the expectation value of $X$, be means of

$$F_X(i) = f_X(i) + F_X(i-1), \quad (11)$$

$$E(Y) = \sum_{i=0}^{\infty} f_Y(i) = \sum_{i=0}^{\infty} \left(\sum_{k=0}^{\min(K,i)} \binom{K}{k} f_X^k(i) F_X^{K-k}(i-1)\right)$$

where either the lower or upper expression may be used for MU-ARQ, but the former is used to derive (13) for SR-ARQ.

The throughput is then, based on the mean number of failures (11) and with respect to $N$ and $K$, given by

$$T(K, N, p) = N / E(Y). \quad (12)$$

When $N \rightarrow \infty$, intuitively $T(K, N, p) \rightarrow p$, but also due to the central limit theorem applied on any of $X_k$. The non-iid reception probability case for $N \rightarrow \infty$ is also straightforward to address as it is simply the worst link that will limit the throughput, i.e. $T = \min\{p_1, p_2, ..., p_K\}$ when $N \rightarrow \infty$.

For the reference case, Multicast SR-ARQ, a similar approach as above is taken. Let $X'$ be another, but discrete geometric distributed, r.v. that instead describes the number of $i$ failures before the first success in a Bernoulli process, with probability $p$ of success on each trial. The cdf and pmf for $X'$ are simply (7) and (8) with $N = 1$. As we are interested to send
a packet until all users have received the packet, we define a second discrete r.v. as \( Y' = \max\{X'_1, X'_2, ..., X'_K\} \). The mean number of failures until \( N \) packets have delivered is simply \( N \cdot E[Y'] \) since each packet round is independent from the other. Inserting (7) and (8) with \( N = 1 \) in (11) for the SR-ARQ case, it can after some calculation be written as

\[
E(Y') = \sum_{m=1}^{M} \frac{M}{m} \left(-1\right)^m \frac{1}{1-q^m}.
\]  

(13)

The throughput \( T'(K, N, p) \) for SR-ARQ is determined analogously to (12), but with \( N \cdot E[Y'] \) replacing \( E(Y) \).

The relative throughput efficiency \( T(K, N, p)/T_{\text{max}} \), where \( T_{\text{max}} = p \), based on (7), (8), (11) and (12) for MU-ARQ is plotted in Fig. 6 for \( K = \{1,2,4,8,16,32,64\} \) users and \( p = \{0.2,0.8\} \) vs. the number of packets \( N \). It is observed that the relative throughput is higher for \( p = 0.8 \) than for \( p = 0.2 \), and that it is above 0.5 in all cases already at \( N \geq 7 \) packets. It is further seen that \( T \to p \) when \( N \to \infty \). The relative throughput for SR-ARQ may also be read at \( N = 1 \) in Fig. 6.

The throughput vs. the reception probability for both multicast MU-ARQ and SR-ARQ are plotted in Fig. 7, where MU-ARQ is plotted for the asymptotic performance \( T = p \) when \( N \to \infty \), and the performance for SR-ARQ is given by (13) for \( K = \{1,2,4,8,16,32,64\} \). Note that the intuitive result that \( T' \to 0 \) when \( K \to \infty \) for \( p \neq 1 \) is seen here, and that the illustrated simulation results are discussed further in IV.A.

## IV. SIMULATION RESULTS

In this section, throughput efficiency, encoding and decoding complexity, header OH (due to signaling of the used weight vector), CP weight distribution, and delay aspects are examined through simulation of the MU-ARQ online algorithm in Fig. 4. Also, the amount of FB OH is briefly addressed.

### A. Throughput Efficiency

Simulated throughput results vs. the reception probability for both ARQ schemes are plotted together with the analytical results in Fig. 7, where \( N = 100 \) and \( N \geq 1000 \) was used in the MU-ARQ and SR-ARQ simulations respectively. For SR-ARQ a simpler and independent simulator was developed and used. Apart from a noticeable perfect match between simulated and analytical results for both schemes, MU-ARQ is also seen to be practically independent of number of users and performs significantly better than SR-ARQ.

### B. En-/Decoding Complexity

The mean en-/decoding complexities regarding additions and multiplications per successfully received CP are shown with one-\( \sigma \) error bars in Fig. 8a-b vs. the number of users respectively. Note that only additions with non-zero elements and multiplications with factors different from one and zero are counted. The simulation parameters used here, and also in section IV.C-D, are \( N = 100, K = \{1,2,3,...,16\} \).

### C. Header and Feedback Overhead

The mean header OH is plotted vs. the number of users \( K \) in Fig. 8c. Given that a sequence number is sent with each packet, the extra OH due to signaling of the weights is here estimated as the entropy of the weights without memory. Practically, the probability of each weight value is determined in the simulation, and the mean entropy for each weight vector per packets is calculated to generate the curves.

The FB OH of the proposed method is assumed identical to multicast SR-ARQ, i.e. scales with \( K \), and is thus not plotted. In this respect, FC is a more efficient method as less FB is required. FB OH reduction remains an item for further studies.

### D. Non-zero Weight Distribution

A set of weight distributions for the CPs is shown in Fig. 9a. One notice that weight values \( \geq 3 \) is never used, and code
weights \{0, 1, 2\}, as suggested by (6), will suffice in practice. It is also seen that on average, there are fewer non-zero weights than the number of users. This depends on that there is a high probability that multiple users have the same rank, and hence require the same code weight to be set to a non-zero value.

E. Decoding and Delivery Latency

The Fig. 4 algorithm is at present not deliberately designed to minimize the packet decoding delay, but improvements are considered. In Fig. 9, arbitrarily chosen trace plots for \( K = 4 \) users exemplifies the packet decoding and delivery instance vs. the packet sequence number. It is seen that packets are delivered in sequence for the user with the highest instant rank of \( W_k \) and that the delivery instance nearly follows the expected rate for \( T = p \). Yet, the rank lagging users’ packets are decoded and delivered first, or just barely before, when all CPs for a block of \( N \) CPs have been received. SR-ARQ on the other hand could deliver packets in transmit order, and does not need to wait to complete the block transmission as for MU-ARQ, but each mean packet transmit time, i.e. 1/\( T' \), is longer.

SR-ARQ would use in total about 36 transmissions in Fig. 9.

V. SUMMARY AND CONCLUSIONS

In this paper, MU-ARQ was extended to multicasting where the core idea was to ensure, through suitable coding and scheduling and a sufficiently large coding field size based on the number of served users, that each user is guaranteed to always receive new rank increasing linear combination of packets at every reception instance. This enabled the least possible number of packets to be sent, i.e. throughput optimal multicasting. A vital difference to prior work [11][12] was that in the proposed scheme, code weights were adaptively selected based on FB rather than adaptively selecting suitable set of user for which a CP was suited to be sent to. In addition, we saw that with the XOR operations, it was not always possible to attain the optimal throughput for larger number of users.

Multicast MU-ARQ and SR-ARQ were analytically and simulative studied regarding throughput for iid channels, proving beneficial for MU-ARQ. Realistic simulations, based on a proposed online coding and scheduling algorithm, were performed and results for en-/decoding complexity as well as signaling OH was presented to illustrate the scheme’s viability, yet challenges with FB OH remains. Future work will consider refined online algorithms, e.g. for reduced decoding latency and weight vector identification complexity, as well as efficient protocols for the non-iid channel scenario.

ACKNOWLEDGEMENTS

The author would like to thank Jonas Kronander at Ericsson Research for valuable discussions.

REFERENCES